IBM

# os/2 Developer

$9.95 U.S.
$10.95 Canada
Display Until
1/1/93

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

Vol. 4 No. 4

## Spotlight on Borland International

## OS/2 at the Indy 500

## Developing Multimedia Applications

24
0 74470 12531 0

# Delivering the Power: WATCOM C9.0/386

- **The Widest Range of 32-bit Intel x86 Platforms**
  32-bit DOS, 32-bit Windows, OS/2 2.0, AutoCAD ADS
- **The Industry's Leading Code Optimizer**
  Advanced global optimizer with new 486 optimizations
- **The Most Comprehensive Toolset**
  Debugger, profiler, protected-mode compiler and linker, 32-bit DOS extender with royalty-free run-time, licensed components from Microsoft SDK, and more
- **The Best Value in 32-Bit Tools: $895***

## Unleash 32-bit Power!

**WATCOM C9.0/386 lets you exploit the two key 32-bit performance benefits. The 32-bit flat memory model simplifies memory management and lets applications address beyond the 640K limit. Powerful 32-bit instruction processing delivers a significant speed advantage: typically at least a 2x speedup.**

### You Get:

- **100% ANSI and SAA compatible:** C9.0/386 passes all Plum Hall Validation Suite tests
- **Extensive Microsoft compatibility** simplifies porting of 16-bit code
- **Royalty-free** run-time for 32-bit DOS, Windows and OS/2 apps
- **Comprehensive toolset** includes debugger, linker, profiler and more
- **DOS extender** support for Rational, Phar Lap and Ergo
- **Run-time compatible with WATCOM FORTRAN 77/386**

**32-bit DOS** support includes the DOS/4GW 32-bit DOS extender by Rational Systems with royalty-free runtime license

- Virtual Memory support up to 32Mb

**32-bit Windows** support enables development and debugging of true 32-bit GUI applications and DLL's.

- Includes licensed Microsoft SDK components

**32-bit OS/2 2.0** support includes development for multiple target environments including OS/2 2.0, 32-bit DOS and 32-bit Windows

- Access to full OS/2 2.0 API including Presentation Manager
- Integrated with IBM Workframe/2 Environment

**AutoCAD ADS** and **ADI** Development: Everything you need to develop and debug ADS and ADI applications for AutoCAD Release 11

**Novell's** *Network C for NLM's SDK* includes C/386

## The Industry's Choice.

**Autodesk**, *Robert Wenig, Manager, AutoCAD for Windows:* "At Autodesk, we're using WATCOM C/386 in the development of strategic new products since it gives us a competitive edge through early access to new technologies. We also highly recommend WATCOM C/386 to third party AutoCAD add-on (ADS and ADI) developers."

**Fox Software**, *David Fulton, President:* "FoxPro 2.0 itself is written in WATCOM C, and takes advantage of its many superior features. Optimizing for either speed or compactness is not uncommon, but to accomplish both was quite remarkable."

**GO**, *Robert Carr, Vice President of Software:* "After looking at the 32-bit Intel 80x86 tools available in the industry, WATCOM C was the best choice. Key factors in our decision were performance, functionality, reliability and technical support."

**IBM**, *John Soyring, Director of OS/2 Software Developer Programs:* "IBM and WATCOM are working together closely to integrate these compilers with the OS/2 2.0 Programmer's Workbench."

**Lotus**, *David Reed, Chief Scientist and Vice President, Pen-Based Applications:* "In new product development we're working with WATCOM C because of superior code optimization, responsive support, and timely delivery of technologies important to us like p-code and support for GO Corp's. PenPoint."

**Novell**, *Nancy Woodward, V.P. and G.M., Development Products:* "We searched the industry for the best 386 C compiler technology to incorporate with our developer toolkits. Our choice was WATCOM."

## WATCOM C9.0/386

▸ 100% ANSI C optimizing compiler ▸ Protected-mode compiler ▸ OS/2 hosted-compiler ▸ Royalty-free DOS extender with VMM support ▸ Licensed components of the Microsoft Windows SDK ▸ Interactive source-level debugger ▸ Linker ▸ Protected-mode linker ▸ OS/2-hosted linker ▸ Profiler ▸ Object code librarian ▸ Object code disassembler ▸ MAKE facility ▸ Patch facility ▸ Object module convert utility ▸ Windows supervisor ▸ Bind facility for Windows applications ▸ 32-bit run-time library object code ▸ Special 32-bit libraries for Windows API ▸ Graphics library for Extended DOS applications ▸ 32-bit Run-time libraries for Windows ▸ 32-bit Run-time libraries for OS/2

MICROSOFT WINDOWS. Version 3.0 Compatible Product

PC MAGAZINE EDITORS' CHOICE September 13, 1988 Watcom C. Version 6.5

INFO WORLD BEST IN ITS CLASS WATCOM C7.0 1989

COMPUTER LANGUAGE PRODUCTIVITY AWARD 1990

COMPUTER LANGUAGE PRODUCT EXCELLENCE AWARD 1990 JOLT COLA

## Special Offer

**Buy *WATCOM C9.0/386* and you'll be eligible to obtain:**

***WATCOM C9.0 Delta Pack*** provides you with the tools necessary to develop and debug 16-bit applications for DOS, Windows, and OS/2. **Only $99.** ($495 comparative separate cost)

***WATCOM FORTRAN 77/386 Delta Pack*** provides you with everything necessary to develop and debug 32-bit FORTRAN applications for extended DOS, 32-bit Windows and OS/2 2.0. **Only $399.** ($895 comparative separate cost)

**WATCOM** **1-800-265-4555**

## The Leader in 32-bit Development Tools

415 Phillip Street, Waterloo, Ontario, Canada
Telephone: (519) 886-3700, Fax: (519) 747-4971
*Price does not include freight and taxes where applicable. Authorized dealers may sell for less.
WATCOM C and Lightning Device are trademarks of WATCOM Systems Inc.
DOS/4G and DOS/16M are trademarks of Rational Systems Inc.
Other trademarks are the properties of their respective owners.
Copyright 1992 WATCOM Products Inc.

# MICROSOFT AND BORLAND STAND BEHIND SMALLTALK/V.

*If you're looking for the fastest, easiest way to develop and maintain applications, see Computerworld. That's where the major object-oriented languages were meticulously rated by actual customers.*

*For reusability of code and ease of maintenance, no product rated higher.*

*The under $500 Smalltalk/V® outranked its $3500 rival from ParcPlace Systems, and neatly outscored C++ from Microsoft and Borland.*

*With ratings like these, it's no wonder a company like IBM really does stand behind us.*

COMPUTERWORLD BUYERS' SCORECARD

| Product | Reusability of code |
|---|---|
| Digitalk's Smalltalk/V | 8.5 |
| ParcPlace Systems' Objectworks/Smalltalk | 8.3 |
| Microsoft's C++ | 7.9 |
| Borland's C++ | 7.3 |

Ratings indicate user satisfaction.

COMPUTERWORLD BUYERS' SCORECARD

| Product | Maintainability of code |
|---|---|
| Digitalk's Smalltalk/V | 8.0 |
| Microsoft's C++ | 7.7 |
| Borland's C++ | 7.6 |
| ParcPlace Systems' Objectworks/Smalltalk | 7.5 |

Ratings indicate user satisfaction.

## IBM TAKES A STAND.

*Because of the leading edge technology and superior quality of Smalltalk/V, Digitalk is now one of only eight companies appointed to IBM's International Alliance for AD/Cycle. Which means Digitalk's Smalltalk/V is a key part of IBM's strategy for the 1990's.*

*It's not surprising, when you consider that our top-rated reusability of code is the essence of object-oriented programming. And that our unequaled ease of maintenance is the feature considered most important by users in the Computerworld survey.*

## DEVELOP FASTER.

*Numerous case histories have shown that you can develop at least twice as fast with Smalltalk/V than with C++. Smalltalk/V is available for OS/2, as well as Windows, Macintosh and DOS. With easy portability across each platform.*

*And every application you deliver is 100% royalty-free.*

*To receive an informative white paper on Smalltalk/V object-oriented technology, call 1-800-531-2344. Please ask for Department 505.*

INTERNATIONAL ALLIANCE IBM AD/Cycle

*If you're looking for the fastest way to develop applications, get behind the winning technology for the 1990's.*

**100% PURE OBJECTS.™**

DIGITALK

# IBM OS/2 Developer

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

FALL 1992


**EDITOR'S COMMENTS**
*Race Cars, Swimming Pools, and OS/2* 5

**SPOTLIGHT**
*OS/2 Tracks The Indy 500* 6
*Borland: Tremor In the Valley* 16

**SOFTWARE TOOLS**
*SourceLink: A Source Code Processor* 30
*Smalltalking With the Database Manager* 36
*Synectics: A New Distributed Processing Architecture* 44
*AM/Workplace: Building Client/Server Applications* 62

**PERFORMANCE**
*OS/2 Productivity Through Multitasking* 58

**MULTIMEDIA**
*Plugging Into MMPM/2* 72
*Networked Full Motion Digital Video* 81
*MMPM/2 and OS/2: The Multimedia Advantage* 90
*Multimedia Waveform Audio Support* 102
*IBM ActionMedia II Support For MMPM/2* 111

**DATABASE**
*Driving Forward with the Database Manager Command Line Interface* 117

**HARDWARE**
*Making OS/2 Run Across Industry PCs* 125

**INTERNATIONAL**
*EMEA Developer Assistance Program Update* 132

**INTERVIEW**
*Bob Orfali and Dan Harkey* 134

The IBM OS/2 Developer is published quarterly by IBM Software Support, Internal Zip 2230, 1000 N.W. 51st St., Boca Raton, Fla. 33429, (407) 982-6408, fax (407) 443-4233. IBM employees and branch office customers can subscribe to the IBM OS/2 Developer through IBM Mechanicsburg's Systems Library Services (SLSS) using the IBM OS/2 Developer's order number: G362-0001. Others can subscribe by calling (800) WANT-OS2. Subscriptions (U.S. only) are $39.95 yearly. International subscriptions are also available by phone (708) 647-5960, fax (708) 647-0537.



© Copyright 1992 by International Business Machines Corp. Printed in U.S.A.

IBM OS/2 Developer
THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

**EDITOR**
Dick Conklin

**EXECUTIVE EDITOR**
Nicole Freeman

**PRODUCTION EDITOR**
Peter Altenberg

**ASSISTANT COPY/PRODUCTION EDITOR**
Lisa Gluskin

---

**PUBLISHER**
Regina Starr Ridley

**GROUP DIRECTOR**
Don Pazour

**NATIONAL SALES MANAGER/ ADVERTISING SALES STAFF**
Cathy Passage
Pam D. Grossman
Michele Anet
*(West: 415-905-2392)*

**REGIONAL SALES MANAGER/ ADVERTISING SALES STAFF**
Veronica Smith
Jo Ben-Atar
Michele Blake
Dave Moreau
*(East: 212-683-9294)*

**MARKETING MANAGER**
Susan McDonald

**ART DIRECTOR/MARKETING**
Christopher H. Clarke

**ADVERTISING PRODUCTION COORDINATOR**
Kathy Bruin

**DIRECTOR OF PRODUCTION**
Andy Mickus

**DIRECTOR OF CIRCULATION**
Jerry Okabe

**CIRCULATION MANAGER**
Kathy Henry

**SUBSCRIPTION INQUIRIES**
U.S.: (800) WANT-OS2
Foreign: (708) 647-5960

*IBM OS/2 Developer* is published by the Personal Systems Line of Business of International Business Machines Corp., Boca Raton, Fla., U.S.A., Dick Conklin, Editor.

Titles and abstracts, but no other portions, of information in this publication may be copied and distributed by computer-based and other information service systems. Permission to republish information from this publication in any other publication of computer-based information systems must be obtained from the Editor.

IBM believes the statements contained herein are accurate as of the date of publication of this document. However, IBM, hereby disclaims all warranted either expressed or implied, including without limitation any implied warranty of merchantability or fitness for a particular purpose. In no event will IBM be liable to you for any damages, including any lost profits, lost savings or other incidental or consequential damage arising out of the use or inability to use any information provided through this publication even if IBM has been advised of the possibility of such damages, or for any claim by any other party.

Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitation or exclusion may not apply to you.

This publication may contain technical inaccuracies or typographical errors. Also, illustrations contained here may show prototype equipment. Your system configuration may differ slightly.

This publication may contain articles by non-IBM authors. These articles represent the views of their authors. IBM does not endorse any non-IBM products that may be mentioned. Questions should be directed to the authors.

This information is not intended to be an assertion of future action. IBM expressly reserves the right to change or withdraw current products that may or may not have the same characteristics or codes listed in this publication. Should IBM modify its products in a way that may affect the information contained in any user of the modification. It is possible, that this material may contain reference to, or information about, IBM products (machines and programs), programming or services that are not be construed to mean that IBM intends to announce such products, programming or services in your country.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. All specifications are subjects to change without notice.

With respect to advertising material herein, this publication does not constitute an expressed or implied recommendation or endorsement by IBM of any particular product, service, company, or technology.

IBM takes no responsibility whatsoever with regard to the selection, performance or use of the products advertised herein. All understanding, agreements, or warranties must take place directly between the software vendors and prospective users.

References to the term OS/2 or OS/2 2.0 are abbreviations for OS/2 Operating System, a registered trademark of IBM Corp.

To correspond with the IBM OS/2 Developer, please write to the Editor at IBM Corp., Internal Zip 2230, P.O. Box 1328, Boca Raton, Fl 33429-1328.

OS/2, CUA, Workplace Shell, PS/2, PS/1, RISC System/6000, Presentation Manager, SQL/DS, SQL/400, AS/400, ES/9000, LAN Manager, and NetView are registered trademarks of IBM Corp.

ABC Wide World of Sports is a registered trademark of the American Broadcasting Corp.

AppleTalk and Macintosh are registered trademarks of Apple Computer, Inc.

Banyan and VINES are registered trademarks of Banyan Systems Inc.

COMDEX is a trademark of the Interface Group Inc.

CompuServe is a registered trademark of CompuServe Inc.

Dorian DATA-1 is a trademark of Dorian Industries Pty. Ltd.

DOS, Windows, XENIX, Microsoft C, and Windows-NT are registered trademarks of Microsoft Corp.

DVI/Digital Video Interactive, 386SX, 486, and ActionMedia are trademarks of the Intel Corp.

Ethernet is a trademark of Xerox Corp.

The Indy 500, Indianapolis 500, Indianapolis Motor Speedway, and the Indianapolis Motor Speedway Hall of Fame Museum are registered trademarks of the Indianapolis Motor Speedway Corporation.

IRIS is the acronym for Integrated Race Information Systems, a collection of application programs written under the OS/2 operating system and used exclusively at the Indianapolis Motor Speedway to organize and distribute race information. It is not an IBM product.

NFS is a trademark of Sun Microsystems, Inc.

Novell Requester, NetWare 3.2, IPX, SPX, and Btrieve are registered trademarks of Novell Inc.

*PC Week* is a registered trademark of IDG Inc.

Prodigy is a registered trademark of Prodigy Services Co.

SourceLink is a trademark of SourceLine Software Inc.

Synectics is a trademark of Parallel PCs Inc.

Turbo Pascal, InterBase, Quattro, QuattroPro, Sidekick, Object Vision, Borland C++, Menus On Demand, Turbo Assembler, Paradox, and dBASE are registered trademarks of Borland International

UNIX is a registered trademark of UNIX Systems Laboratories

Wavefront is a registered trademark of Wavefront Technologies

Acer America, Advanced Logic Research, Apricot Computers, AST Research, AT&T Data Systems Group, Club America Tech., Commodore International, Compaq Computer, CompuAdd, CSS Laboratories, Dell Computer, Digital Equipment Corp., Dtk Company, Epson Portland, Everex System, Grid Systems, Hewlett Packard, Intel, Leading Edge Productions, Memorex Telex, Mitac International, Modern Computer, NCR, NEC Technologies, Netframe Systems, Northgate Computer Systems, Olivetti Advanced Tech., Parallan Computer, Reply, Siemens Nixdorf International, Tandon, Tandy Electronics, Tatung Co. of America, Toshiba America, Tricord Systems, and Wyse Technology are registered trademarks of their respective companies.

# Editor's Comments

A cover shot of the Indianapolis 500? What's that got to do with OS/2™? OS/2 applications are popping up in some interesting places—Cape Canaveral, the Johnson Space Center in Houston, and even at the Indy 500. While this isn't exactly your run-of-the-mill business application, it's proof that OS/2 continues to be used in ways its designers probably never imagined. Patrick Karle, Lenny Zwik, and Pete Woytovech take us behind the scenes and explain the intertwined histories of IBM and the "Five Hundred Mile Race." Kind of gives new meaning to the term "mission critical."

And while we're talking about creative OS/2 applications, have you given much thought to multimedia? We're continuing our series with several articles on OS/2's Multimedia Presentation Manager/2™ (MMPM/2) product and its companion Toolkit. MMPM/2 brings a new 32-bit media control interface to OS/2 that shields applications from underlying hardware and simplifies programming, while Digital Video Interactive technology brings audio and full motion video to LAN-based applications.

Our Spotlight this issue focuses on Borland International, a software company with consistently award-winning products and driven, committed employees. Lynn Lathrop went inside the company and interviewed its people about teamwork, quality, and a new swimming pool.

Lots of other changes are underway. Our Tools columnist, Brian Proffit, is leaving us for a new position as Director of *PC Week* Labs. Congratulations Brian! In our next issue, we'll introduce a new columnist, Sam Henderson, whom many of you already know from the various bulletin boards, forums, and CompuServe™. Sam's Q & A column will answer your OS/2 technical questions. Just "Ask Sam."

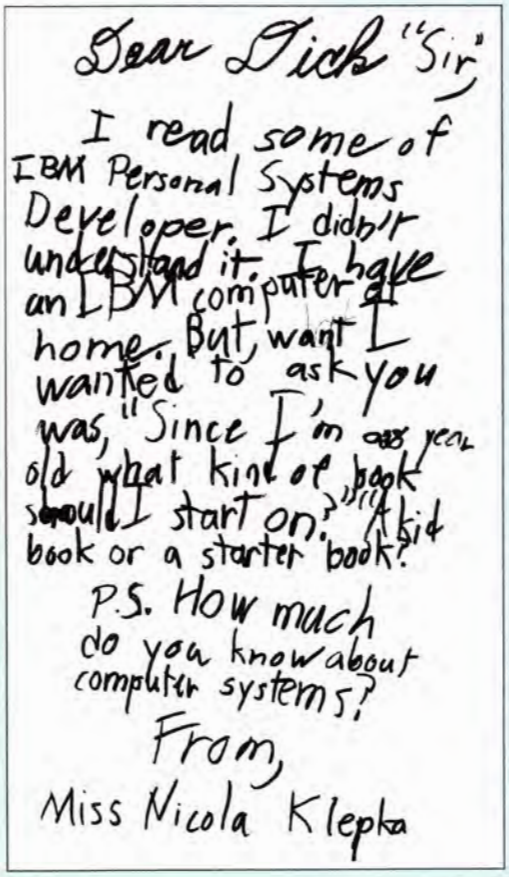
By the time you read this, we will be participating in a new *OS/2 Developer* magazine section on CompuServe. You can talk to your editor, article authors, columnists, and other readers. We're excited about this new way to get feedback and suggestions from our readers. Drop by and say hello. Look for us in the OS2DF2 forum.

Going to COMDEX? See you at the IBM exhibit!

Dick Conklin

*Dick Conklin*

## LETTERS TO THE EDITOR



Dear Dick "Sir"
I read some of IBM Personal Systems Developer. I didn't understand it. I have an IBM computer at home. But, want I wanted to ask you was, "Since I'm 8 years old what kind of book should I start on? A kid book or a starter book?
P.S. How much do you know about computer systems?
From,
Miss Nicola Klepka

Dear Nicola,

Thanks for your nice letter. Learning about computers can be quite a challenge, but it helps to have an IBM PS/2™ at home! I'm sending you a book my daughter Michelle read several years ago when she took a computer class at school. She recently finished college and started student teaching. Soon she will be teaching other kids how to use computers.

Keep on studying and using your computer. Perhaps in a few years you will be writing computer programs for other people to use. Who knows—someday you may even write for this magazine!

Best wishes,
Dick Conklin

Spotlight

# OS/2 Tracks The Indy 500

*Patrick Karle*

*Lenny Zwik*

*by Patrick Karle, Lenny Zwik, and Pete Woytovech*

Race cars weren't the only fast machines at the 1992 Indianapolis 500™.

For the first time, IBM PS/2s and the Integrated Race Information System (IRIS), using OS/2 Extended Edition Database and Communications Managers, were able to place timing, scoring, and administrative data at the fingertips of race officials, press, and fans. A newly installed 16-megabit token ring network tied a 33MHz PS/2 Model 95 database server in the scoring tower to PS/2 Model 57 SLC workstations installed throughout the Indianapolis Motor Speedway™.

Two additional PS/2 Model 95s collected and processed timing data generated by radio frequency transmitters on the Indy cars. The cars passed over a series of 11 loop antennas embedded in the track surface around the 2.5-mile race course. Four additional antennas monitored cars in and out of the pitlane and garage area, as shown in Figure 1.

IRIS was developed to store and retrieve entry, registration, and technical inspection data for the United States Auto Club (USAC, the nonprofit motorsports organization that sanctions the Indianapolis 500) and to integrate these administrative functions with timing and scoring information. At the IBM lab in Boca Raton, Fla., Lenny Zwik, Pete Woytovech, and Bill Bodin, the IRIS development team, worked with Indianapolis Marketing branch office systems engineers Bob Cole and Dave Berryman to learn the details of auto racing and the Indy 500 to meet the technical needs of USAC and the Speedway.

## IRIS AT THE INDY 500

Lenny Zwik, leader of the IRIS development team, explains that a key objective of IRIS was to migrate the processing of entry, registration, and technical inspection data from a paper-based to an electronic system. USAC and IMS officials, in consultation with IBM's special marketing projects team, decided that OS/2's multitasking capabilities would best serve race officials who needed quick, often simultaneous access to race information. Another IRIS objective was to design an OS/2 application that could process information generated by the Dorian DATA-1™ radio transmitter-based automatic timing and scoring system, operated by the USAC timing and scoring team. The Dorian DATA-1 was installed at IMS in 1990.

The DATA-1 is the first fully automatic car scoring system used at the speedway, says Art Graham, the Speedway's director of timing and scoring. Designed, developed, and manufactured by Dorian Industries of Melbourne, Australia, DATA-1 is guaranteed to an accuracy of 1/10,000th of a second at speeds up to approximately 400 miles per hour.

The DATA-1 system uses small (two by five by three inch) radio-frequency transmitters fixed to the floor of each car and loop antennas called "time lines" embedded in the track. Each transmitter sends out a constant digital identification signal. As the cars pass over a loop antenna, the transmitter electronically identifies itself by number. The antenna picks up the signal and feeds it to one of the DATA-1 trackside recording computers, or TRCs. Each TRC, housed in a weatherproof box mounted inside the retaining wall, performs an analog to digital conversion and time

stamps the crossing. A TRC will store data for all cars crossing an antenna during a race or practice session. A PS/2 Model 95, serving as the DATA-1 manager, collates and organizes timing events from all TRCs. It then transmits its event record file to another PS/2 Model 95 which, in turn, executes the IMS/USAC Scoring Manager applications. These applications, written by USAC programmers, generate an event log of chronologically ordered timing and scoring records. Each record identifies a car, the antenna and time line it crossed, and other, more detailed information such as trap speeds.

The Scoring Manager PS/2 relies on a custom program running on an IBM PortMaster/2 card to control communication traffic to and from the DATA-1 Manager PS/2. The PortMaster™/2 is used to generate five output data streams from the Scoring Manager, including a standings summary, scoreboard driver, and event log.

The event log is transmitted to various locations around the Speedway via asynchronous protocols running at 19.2 kilobaud. The Scoring Manager uses this information to create daily and monthly practice and performance histories for each driver and car combination. Performance histories are transmitted to 50 video monitors located in the press box, the ABC Wide World of Sports™ broadcast booths, and other locations around the track. The Scoring Manager also provides data so the scoring pylon control computer can update car numbers on the scoring pylon, where spectators can see moment-to-moment standings.

Unlike photo electronic or infrared timing systems, the IMS/USAC DATA-1-based

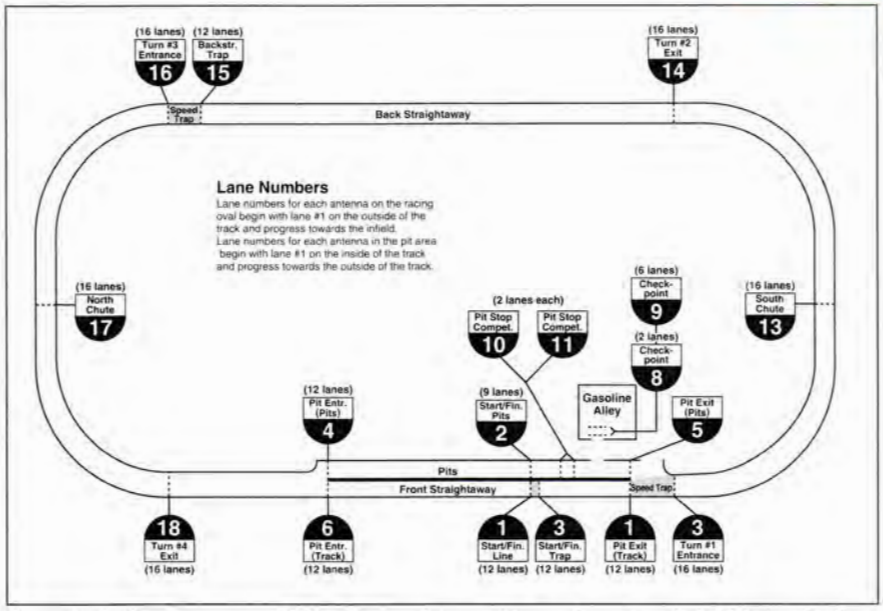



*Figure 1 : As the cars pass over the time lines, each transmitter sends a unique digital ID signal to the antennas (Diagram, Patricia Navarro)*

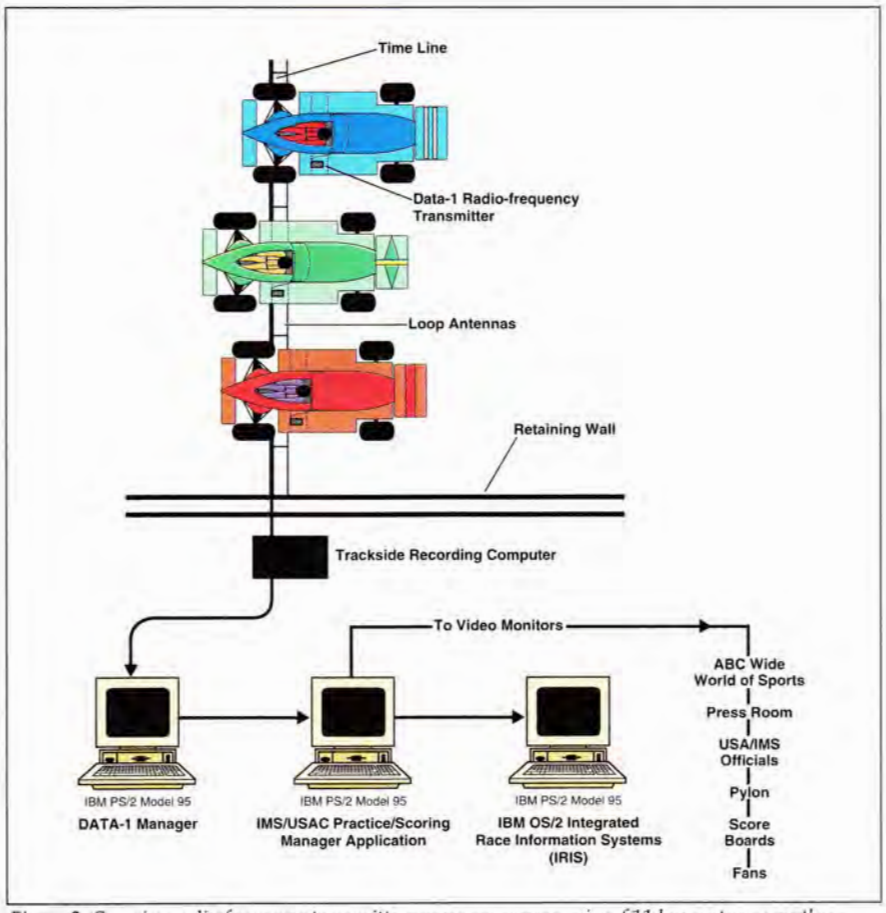



*Figure 2: Carrying radio-frequency transmitters, cars pass over a series of 11 loop antennas as they circle the Speedway track (Diagram, Ryan Hoover)*

system is sophisticated enough to register more than one high-speed race car passing the same time line simultaneously. Each time line electronically divides the track surface into 12 to 16 lanes, like a running track. Because two cars cannot cross the same lane at the same time, the TRC is able to pick up a separate transmitter signal and time for each car. This system is shown in Figure 2.

*A racecar speeds around the Indy track*

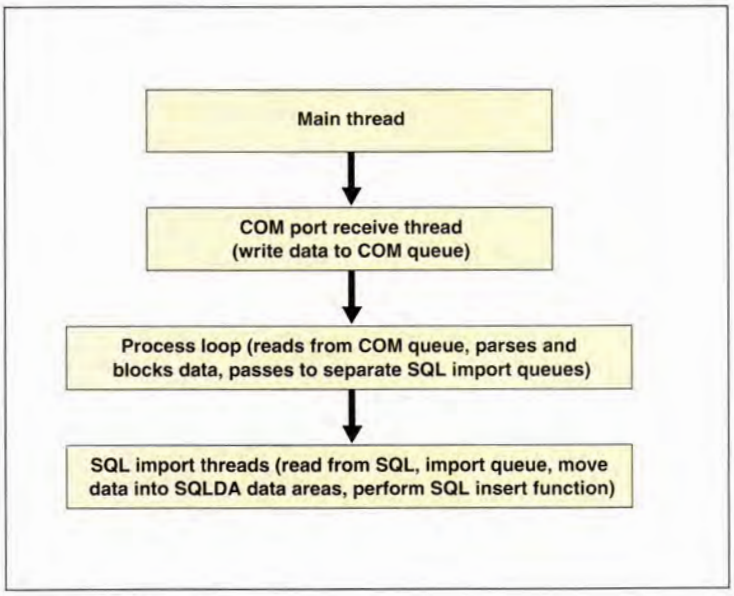



*Figure 3: INDY Importer layout*

The INDY Importer is a native OS/2 application written entirely in C (specifically in Microsoft C™ v. 6.00a). In this multithreaded application, some threads run at very high priority levels. As data collection and manipulation and SQL procedure are not inherently visual processes, the application is an OS/2 windowable text application, rather than a PM application.

## PROGRAM LAYOUT

The program is laid out as shown in Figure 3. The main thread opens all queues, allocates all thread stacks, and starts all SQL import threads (there can be from one to 22 import threads, each dedicated to a specific data record type, allowing for addition or deletion of individual record types from the database). It then loops the data from the COMM queue, parses the data by record type, qualifies it for completeness (each record is a fixed known length), and passes it to an SQL import queue to be imported by the correct thread.

The COM thread reads from the COMM port, since the data feed is a 19.2K bps unhandshaken input from a broadcast output, and writes this data to a 64K queue, to prevent data overrun by the input. This thread runs at time-critical priority.

The SQL import threads (one for each record type), read from their respective queues, convert the data elements into the correct representation for SQL use (field in an SQLDA data area), then perform an `SQL INSERT` statement (prepared during compilation). Since the Database Manager handles the record and file locking, there is no need to single-thread the `INSERT` function internally.

## IRIS AND INFORMATION DISTRIBUTION

IRIS helped the timing and scoring team improve the distribution of information such as the fastest lap of the race, current serial scores and standings for any lap or series of laps, elapsed time of any car on any lap, current aggregate pit time for all cars within 10 laps of the leader, and pit times for every car in the race—to name just a few available functions. IRIS also allowed the USAC auditing team to pass judgement on rule infractions as they were reported.

# No Matter What Your GUI Destination

# CASEWORKS

# Can Take You There

**Now You Don't Have to Worry About Choosing the Wrong GUI Development Path...CASEWORKS Lets You Easily Change Direction — by Switching Languages or Platforms — Without Wasting Any Time!**

All of the choices surrounding Graphical User Interfaces (GUIs) can make picking a GUI development path seem impossible. Deciding between Windows™ or Presentation Manager™ (PM) is only the first step toward building client/server applications. You need a tool that lets you generate code to Windows or PM for a variety of languages and APIs, and that lets you migrate interfaces among languages and platforms if you change your mind. *CASE:W™ and CASE:PM™ are the only tools that give you that flexibility.*

**CASE:W for Windows Development**

Windows developers face the challenge of migrating from C to C++ and class library programming. CASE:W makes it easy by generating source code for the three "standards" of Windows programming: the Windows C API, Microsoft Foundation Classes™ and Borland's ObjectWindows™.

Simply buy CASE:W along with a "Knowledgebase" for C, or C++ and a class library. Use the visual designer to create and test your interface. Then generate expert-level, tightly-written, thoroughly-documented code from your knowledgebase. To use the interface in a different environment, simply buy that knowledgebase and generate again. You can even migrate the interface to PM.

**CASE:PM for Presentation Manager Development**

When you're developing "mission-critical" applications for OS/2™2.0, you need a strategic tool that can build PM applications for both 16- and 32-bit PM environments. Our new CASE:PM VIP tool gives you a head start on building client/server applications, and lets you easily migrate legacy applications to OS/2 2.0. Plus, you can leverage all the advanced capabilities of CUA '91.

**The Safest Development Route**

With CASEWORKS, you get the security of using the leading GUI development tools. Microsoft® includes a version of CASE:W with its QuickC™ for Windows, and more corporations use CASE:PM for developing strategic PM and client/server applications than any other standard language tool. Both tools generate code without restrictions, letting you add any code, anywhere. Plus, a regeneration facility preserves your changes. And *you* control the generated code, with no proprietary languages, runtimes or licensing fees.

Don't avoid making a decision because you're uncertain which way to go. Choose the only company that can take you to Windows or PM, through any development route: CASEWORKS.

CASEWORKS™

1 Dunwoody Park, Suite 130, Atlanta, GA 30338
**1-800-635-1577** • (404) 399-6236 • Fax (404) 399-9516

It was important that IRIS integrate entry, registration, and technical data into an accessible yet secure database. Participation in the Indianapolis 500 is by invitation only, and IMS sends invitations only to previous entrants and to parties with viable cars who request an invitation. These invitations and the resulting entries generate a tremendous amount of information. A significant percentage of that data, such as car numbers, owners' and sponsors' names, chassis and engines, driver names and biographical information, is needed by administrative departments outside of timing and scoring.

When each team is entered in the race, it must be registered, fees must be paid, and all teams' personnel must hold USAC licenses and racing credentials. Each car must pass an extensive series of technical inspections that range from certifying the structural integrity of the cars' mechanical components to making sure numbers on the cars can be read by USAC officials at speeds of over 200 m.p.h.

Until 1992, registration information was collected on various forms. IRIS represents the first time in the 81-year history of the race that a computer system has been used to integrate the information. When USAC and IMS chose to use OS/2, only OS/2 Extended Edition facilities were used to develop IRIS. Except for the import program, the system uses only Database, Query Manager, and Remote Data Services on a LAN.

The IRIS user interface was developed as a series of menus and panels. Each of these objects runs various Query Manager procedures and queries to access the database. For security, the developers instituted a combination of User Profile Management services and queries that access a table in the database and describe access levels for each active logon ID.

Chief steward Tom Binford, who serves as the chief umpire and head coach of the Indianapolis 500, said that he was very impressed with the increased capability that the system gives his officials to spot infractions.

## THE 1992 RACE

On May 9, Roberto Guerrero drove his Quaker State Buick-Lola to a four-lap qualification record of 232.482 miles per hour, marking the need for speed and accuracy in timing, scoring, and officiation. With more than seven million dollars at stake, calculating speeds faster than the blink of a human eye is not just an exercise in theoretical physics.

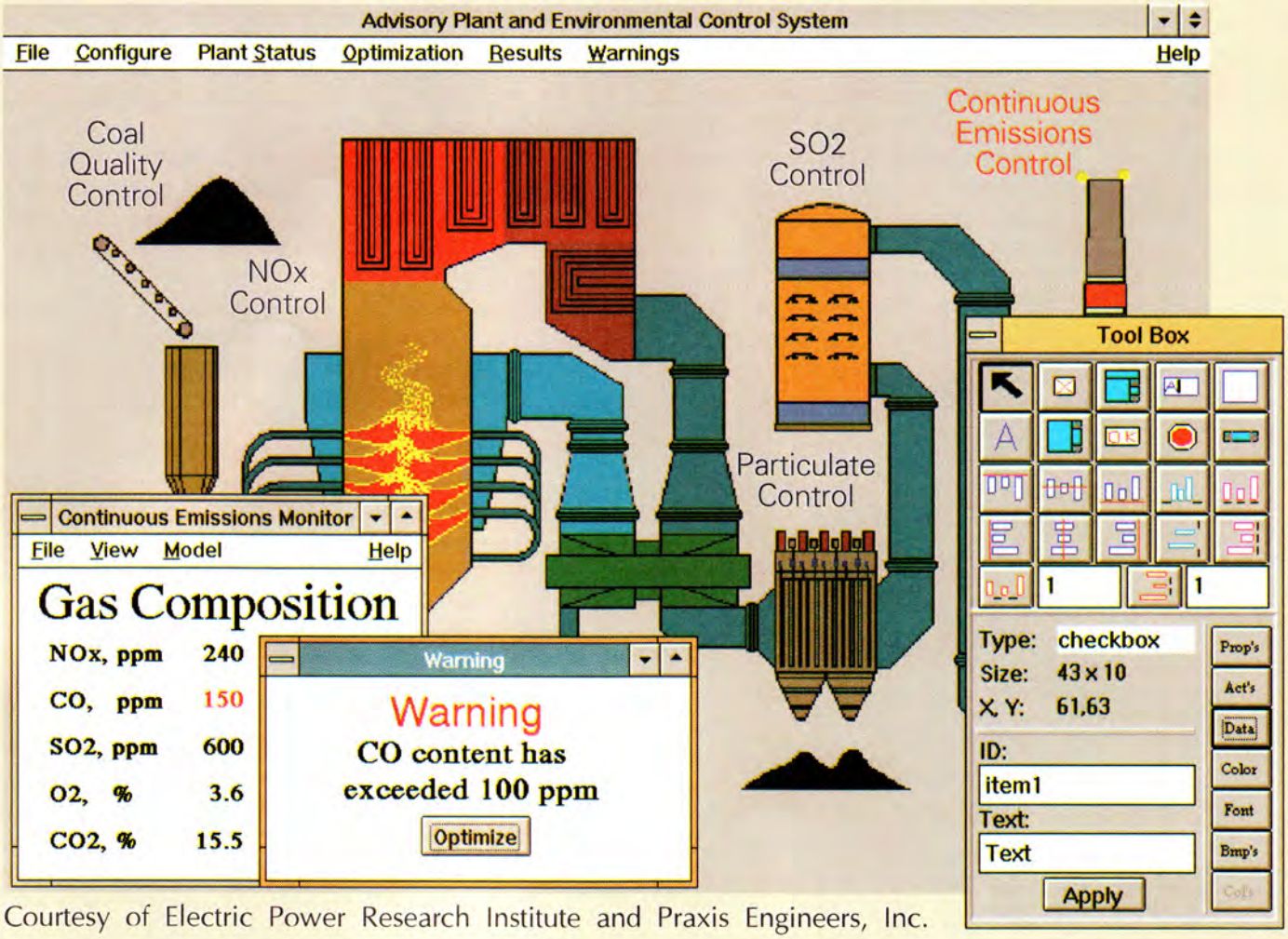
This year's race ended in the closest finish ever as Al Unser, Jr. edged out Scott Goodyear to win his first Indy 500. As each of the two drivers battled to put his car ahead in the final hundred yards, the finish appeared almost a dead heat. Unser's and Goodyear's transmitter signals were picked up by loops embedded at the finish line and collected by IBM PS/2s. As the two racers approached the Victory Circle platform, USAC officials were able to determine the split between the first and second place finishes and to announce the unofficial results to spectators and the press. The slim margin of .043 seconds cost Scott Goodyear well over half a million dollars; Unser won $1,244,184 to Goodyear's $609,333.

USAC president Richard King commented, "This is a big step toward our goal of releasing the official race results in no more time than it takes to complete the technical inspection of the winning car."

## IBM'S 65 YEARS AT THE INDY 500

The IBM PS/2 with OS/2 is the latest in a long line of IBM machines to be used at the Indianapolis 500.

# GUI Tool for Professional Developers




Courtesy of Electric Power Research Institute and Praxis Engineers, Inc.

So you'd like to develop serious applications, but you haven't found your ideal GUI tool for OS/2?

GUILD is for YOU!

- ***Productive.*** Point & click screens, actions and data.
- ***Flexible.*** Open architecture for custom C/C++ coding.
- ***Portable.*** Port across Windows, OS/2, Motif and Mac.

(415) 593-3200
Phone

(415) 595-8158
FAX

GUILD
PRODUCTS INCORPORATED

1301 Shoreway Road, Belmont, Calif. 94002

Trademarks and registered trademarks belong to their respective holders.

In fact, 1992 marks 65 years of continuous use of IBM equipment in the Speedway's timing and scoring operation. IBM has lent support to the race since 1927, when the company's founder, T.J. Watson, leased key punches and tabulators on a one-day basis to Chester S. Ricker, the Speedway's first director of timing and scoring.

*Figure 4: In 1956, scorers wrote car numbers on IBM cards; keypunch operators then punched time codes into the cards for post-race processing (photo courtesy Raymond House)*

*Racecar teams chose IBM PS/2 equipment for use in the Indy 500 pit*

In those days, timing and scoring was located on the second floor of the old pagoda. The 33 car scorers, many of whom were IBM employees and customers, sat at long tables with their IBM cards and time clocks, writing the cars' times on the cards. A photo from that era is shown in Figure 4. Keypunch operators punched the time codes into the cards after the race. In 1956, USAC began mark-sensing, encoding times onto the cards with lead pencils. This system speeded up the post-race processing sessions at the IBM office in downtown Indianapolis—sessions that often took all night and part of the next day!

In 1964, USAC migrated Ricker's data processing scheme from IBM punched-card tabulators to an IBM 709 mainframe computer, taking advantage of new magnetic tape and stored-program technology. In 1982, IBM PCs were incorporated into the timing and scoring scheme, running programs written in BASIC that automated the numbers on the scoring pylon.

In 1990, IMS and USAC made an important agreement with IBM, making IBM the official supplier of information processing equipment and the PS/2 the official computer of USAC and the Indianapolis 500. That year, IMS and USAC installed the Dorian DATA-1, attached to a network of PS/2s acting as data managers.

Robert D. Lohman, manager of IBM's hardware marketing support for IBM Personal Systems Line of Business, says that IBM's participation with USAC and the Indianapolis Motor Speedway offers an opportunity to introduce new technology in a highly visible arena and to demonstrate the value of the IBM PS/2 and OS/2 to prospective users.

In 1990, IBM donated an exhibit entitled "The History of Timing and Scoring" to the Indianapolis Motor Speedway Hall of Fame Museum. The permanent multimedia exhibit tells the 81-year story of the people and machines that timed the world's best-known auto race. That year, IBM also instituted the Fastest Lap of the Race award of $10,000, which goes to the driver who turns the fastest lap of the race. The 1992 winner was Michael Andretti, with a speed of 229.118 m.p.h. on lap 166.

## OS/2 AT THE 1992 SUMMER OLYMPICS

The Indianapolis 500 is not the only sporting event to choose OS/2 for precise information storage and retrieval. Nearly 100,000 people at the 1992 Summer Olympics in Barcelona, Spain, depended on the vast Olympic Information System based on two IBM 9021 mainframe host computers and sent to more than 4,400 IBM PS/2 workstations running OS/2. More than 11,000 athletes, 11,000 media representatives, 40,000 members of the "Olympic Family," and 30,000 volunteers relied on the system to support their needs.

The Results Information System, with 400 networked PS/2s, was installed at each sports venue. It allowed the manual or automatic input of competition results and provided those results in varying formats, such as a direct computer link to press agencies or a TV data feed to broadcasters.

The Olympic Information and Communication System, based on 2,000 networked PS/2s, provided journalists, judges, athletes, and volunteers access to general information such as event results, schedules, medal standings, press releases, and weather reports. It also provided e-mail services for Olympic family members and the press.

The Commentator System, based on 1,100 networked PS/2s, allowed broadcast commentators to access event information or athlete biographies with a color touch screen. According to event organizers, the system was a vast improvement over previous Olympics at which commentators could access only a few information channels at a time.

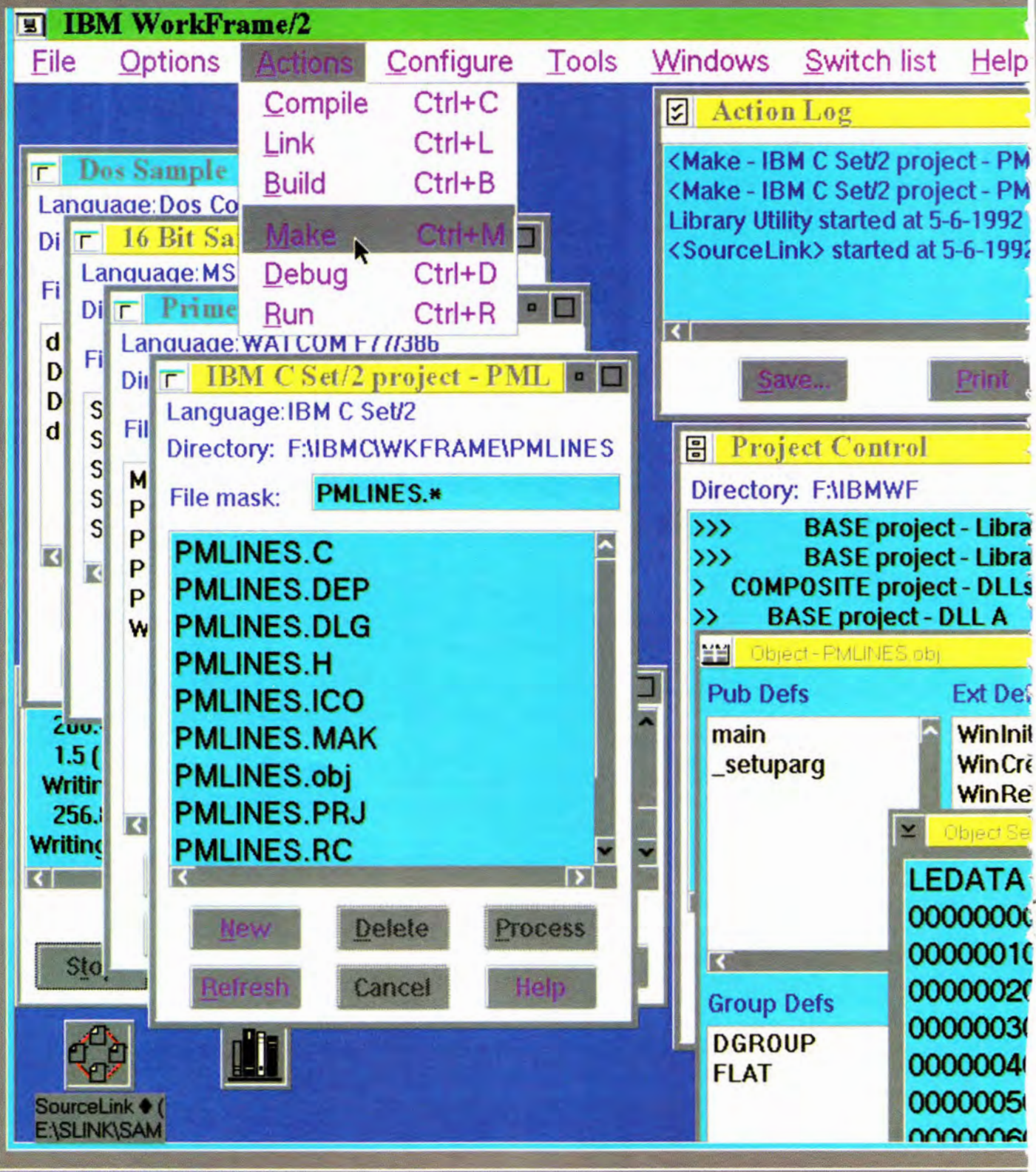
The Operations Management Information System, based on an AS/400 midrange computer with 350 networked PS/2s, was used by Olympic employees for managing accommodations, ticketing, transportation, and medical services.

Innovations of the Olympic Information System included instant availability of data and results from all sites, immediate transmission of unofficial results, and automatic identification of the languages in which users need to receive information.

**Patrick Karle**, *Patrick Karle Associates, 2467 Rt. 10 East #34-5A, Morris Plains, N.J. 07950. Karle is managing director of his own marketing communications firm, specializing in publicity for IBM PS applications. His campaigns promoting IBM PS/2s at the Indy 500 won the Pyramid Award for excellence in public relations in 1991 and 1992. Karle also teaches at the Stillman School of Business at Seton Hall University.*

**Lenny Zwik**, *IBM Corp., 1000 N.W. 51st St., Boca Raton, Fla. 33429. Zwik is a development engineer and manager in the setup development and problem determination department of IBM Engineering Software. He has held positions in manufacturing, engineering, and OS/2 development. Zwik is also a U.S. Auto Club official; he serves as deputy chief of IRIS systems at the Indianapolis Motor Speedway.*

**Pete Woytovech**, *IBM Corp., 1000 N.W. 51st St., Boca Raton, Fla. 33429. Woytovech is a staff programmer and member of the IBM Engineering Software problem determination team. He currently works on solving customer problems. Woytovech is also a U.S. Auto Club official, serving as a member of the IBM systems staff at the Indianapolis Motor Speedway.*




OS/2

*In Canada, call 1 800 465-7999. HelpWare limited to U.S.A. only. IBM and OS/2 are registered trademarks and OS/2 Crash Protection and HelpWare are trademarks of International Business Machines Corporation. Windows is a trademark of Microsoft Corporation. CompuServe is a registered trademark of CompuServe, Inc. © 1992 IBM Corp.

# To a software developer, this is what heaven looks like.

Most people wouldn't know what to make of a screen like this. But developers like you know a screen like this can help make all kinds of applications. With OS/2® 2.0, you can develop the DOS, Windows,™ OS/2 and host-based apps end users need. And you can do it faster and easier than ever before. Because OS/2 2.0 can make the most of your 386 or 486 processor.

Now you can edit in one window, compile in another, profile in a third and test in a fourth. Preemptive multitasking makes everything run smoothly and responsively. And OS/2 Crash Protection™ helps shield running applications from each other, so if one goes down it won't affect the others. Instead of rebooting, you just restart the affected app and continue. And since OS/2 2.0 is a 32-bit operating system, programs are easier to write and run faster, too. Which all adds up to improved productivity and reduced development cycle time.

But maybe the best part is that for less than the cost of DOS and Windows, OS/2 gives you a whole lot more. And to keep your cycle rolling, a full range of services and support are available, like on-line help through OS/2 Support line, Bulletin Board and IBM Link. Or you can join the IBMOS2 and OS2DEV conferences on CompuServe,® where you can meet IBMers, users and developers who can find fast answers to your questions. For an IBM authorized dealer near you, or to order OS/2 2.0 from IBM, call **1 800 3-IBM-OS2.***

- *OS/2 Crash Protection helps shield applications from each other.*
- *The integrating platform of choice for DOS, Windows and OS/2.*
- *Preemptive multitasking for responsive, reliable execution.*
- *32-bit flat address space for productive programming.*
- *A full range of IBM services and support.*

IBM®

Spotlight

# Borland: Tremor In the Valley

*Lynn Lathrop*

*by Lynn Lathrop*

In 1983, a young French mathematician decided to make his mark on the world. Philippe Kahn founded a company called Borland in Scotts Valley, Calif., and a year later began selling Turbo Pascal™ software by mail. Started by one man with a vision, the company has expanded to employ almost 2,000 people. In December 1991, Borland broke ground for a 33-acre, campus-style headquarters in Scotts Valley. Its product line has grown from the highly successful Turbo Pascal to include the Paradox™, dBASE™, and InterBase™ database products, the Quattro Pro™ spreadsheet, the Sidekick™ personal organizer, and programming products such as ObjectVision™ and Borland C++™.

Borland maintains European headquarters in Paris, with subsidiaries in London, Frankfurt, Amsterdam, Madrid, Milan, Stockholm, and Copenhagen. Asia/Pacific subsidiaries exist in Tokyo, Hong Kong, and Sydney, with sales offices in Korea, Malaysia, and Taiwan. The company also runs a manufacturing facility outside of Dublin, Ireland.

The guiding principles of this flourishing company are "to provide software craftsmanship, to be the best of breed, and to be a champion of the customer." Borland's mission is "to empower the user through 'software craftsmanship,'" a watchword of founder Kahn. The key to software craftsmanship at Borland is the use of object-oriented programming technology. Simply put, "we look at things as objects," says Pat Brogan, director of business development.

Kahn believes that object-oriented programming is to software today what the integrated circuit was to hardware 20 years ago: the central enabling technology that fuels a new approach. Object-oriented programming is based on the concept that developers group code and data into "objects," then use these objects to build applications instead of building them from scratch. Object-oriented programming allows developers to recycle much of their code; objects may be reused and recombined in a variety of ways, reducing the need to reinvent code to accomplish new tasks.

At Borland, this high degree of reusability reduces development time and cost. Object-oriented code is modularized, meaning that it has been tested, used, and checked multiple times. Borland's average development cycle is six to nine months between releases, compared to an industry average of 12 to 18 months.

## TEAMWORK AT BORLAND

Teamwork also has something to do with the shortened development cycle. Paul Gross, general manager of the Application Development Business Unit, explains, "It's been a Borland philosophy for a long time to use small teams. The overall number of people on a project may be large, but the team size on a given stand-alone piece is five maximum. We're set up so our basic structure is a research and development project. Teams

report to one research and development project leader who owns the overall project, and then there are several project leaders under that person. In a sense, it's management by peers, not by organization."

As a member of the research and development team for ObjectVision, Bill Turpin feels that being on a small team is "more like playing in a small band instead of a large band. You have only one person playing each instrument, and that one person has full responsibility for carrying his or her part of the song. You have to work together and improvise a little bit more—but the productivity is so much higher."

*Pat Brogan*

Individual teams are "co-located," a company practice that grew out of the 1989 Loma Prieta earthquake, the epicenter of which was near Scotts Valley. The earthquake hit two weeks before Quattro Pro was to ship. As one Borland building was condemned, the company moved all its employees into the MIS area. "It was a great environment," explains Gross. "Quality Assurance would come up with a bug, research and development was right there to look at it, and the publications people put the information into the readme. That was a big success for Borland that came out of devastation." The company is still set up in one space, which increases communication and productivity. Turpin says that "the co-location concept really fosters team spirit. It's hard for a problem to exist without everybody knowing it. It's a very good way for a team to work."

*Paul Gross*

But what about the downside of using small teams, the fact that most research and development teams work long, hard hours? Borland takes care of that by providing an environment conducive to high-energy work and play. The company has an on-site recreation center with a full set of exercise equipment, where an aerobics champion teaches standing-room only classes. The gym is open 24 hours a day, the cafeteria is open nights, and ping-pong tables are scattered throughout the work areas. "Nobody cares if you go to play tennis in the middle of the

day," says Brogan. "We have very good people. Our new facilities will have a swimming pool because management lost a bet that the development team couldn't get Quattro Pro out in six months."

## DEVELOPING FOR OS/2 2.0

ObjectVision and Borland C++ for OS/2 2.0 (the latter release planned for the end of 1992) are two products those "very good people" have concentrated on bringing to the marketplace. "Borland is pleased to bring its development tools to the OS/2 platform," says Paul Gross. "Borland believes that 32-bit technology will meet the needs of programmers worldwide." These programmers include the development teams at Borland. "We spent some time interviewing our developers before doing the OS/2 2.0 launch where we presented our views as a developer and as a user of OS/2," explains Gross. "What they said in terms of developing

for OS/2 2.0 over Windows™ is that it was really great. Sure, there were rough edges, but there were lots of really strong comments about how the API is better than Windows. It's so much more intuitive once you learn a certain class of functions that are generalizable. Other classes of functions, like handling and graphics, are so much better. Plus, people found that the on-line help was very good."

For the Borland teams, according to Gross, coding for 32-bit was not difficult. The development group has people with 16- and 32-bit backgrounds. Much of the development is done in C or C++, so developers are shielded from many of the issues of 16- vs. 32-bit for most of the process.

*Alex Lane*

The developers also appreciated coding for flat model. Alex Lane, product manager for Borland C++, notes, "One day I went and asked the developers, 'What, for you, is the one critical thing about OS/2?' The reaction was almost like a Broadway show: heads popped over cubicles and out of offices chorusing, 'No more segments.'"

Gross says, "If there's one comment I get most from developers, it's that OS/2 is a *real* operating system. It's 32-bit and it uses multitasking—it's preemptive. And when a task crashes, the operating system doesn't go down."

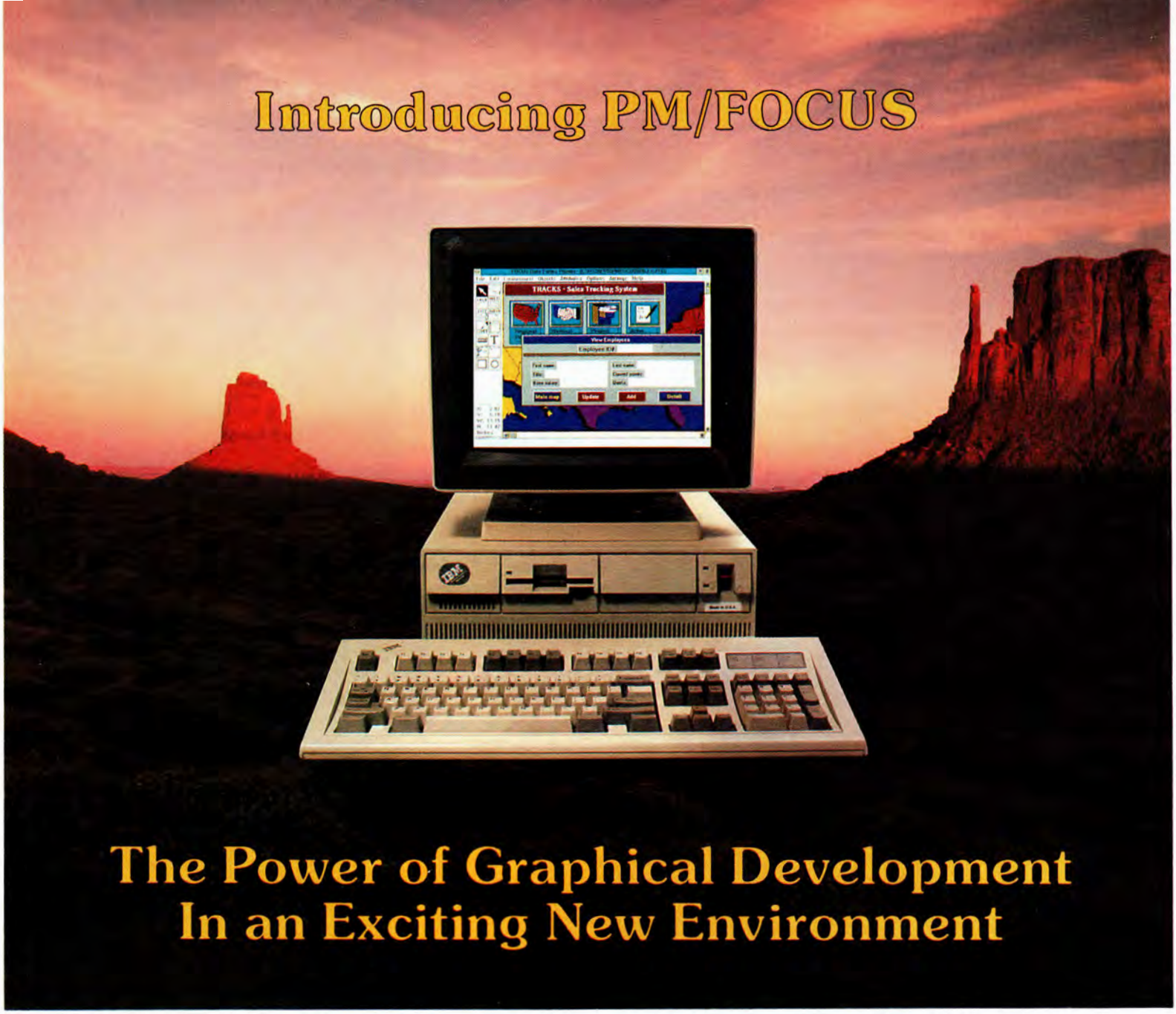
"We took full advantage of the 32-bit environment in terms of capabilities like multithreading and increased performance" says John Ho, director for compiler, debuggers, and application framework development, who coordinated Borland C++ for OS/2. "We don't have to do segments anymore, we don't have selectors to worry about—we can really take advantage of that 32-bit address. OS/2 is a very good platform...there's tremendous opportunity there for applications that were limited by the 640K barrier and speed. There's an opportunity to do some new, bigger applications. We expect to see applications, including our own, become significantly faster in compilation-intensive areas."

*John Ho*

Borland would like developers to use ObjectVision and Borland C++ for OS/2 to develop those applications. ObjectVision is an application development tool that allows nontechnical users as well as developers to create interactive applications without programming. When ObjectVision was presented at the OS/2 Tools Conference in San Francisco last spring, program manager for ObjectVision Carroll Hall noted that "in general...the audiences I spoke to had very little previous knowledge of ObjectVision, but they were very impressed with its functionality after seeing it demonstrated."

*Carroll Hall*

# Introducing PM/FOCUS




## The Power of Graphical Development In an Exciting New Environment

Over three years of development and testing have transformed a vision into reality. An ideal development environment created by the perfect harmony of proven database technology within an exciting new visual landscape. Information Builders introduces PM/FOCUS, a graphical development and decision support system combining unparalleled data access and reporting with the versatility, stability and power of OS/2 2.0.

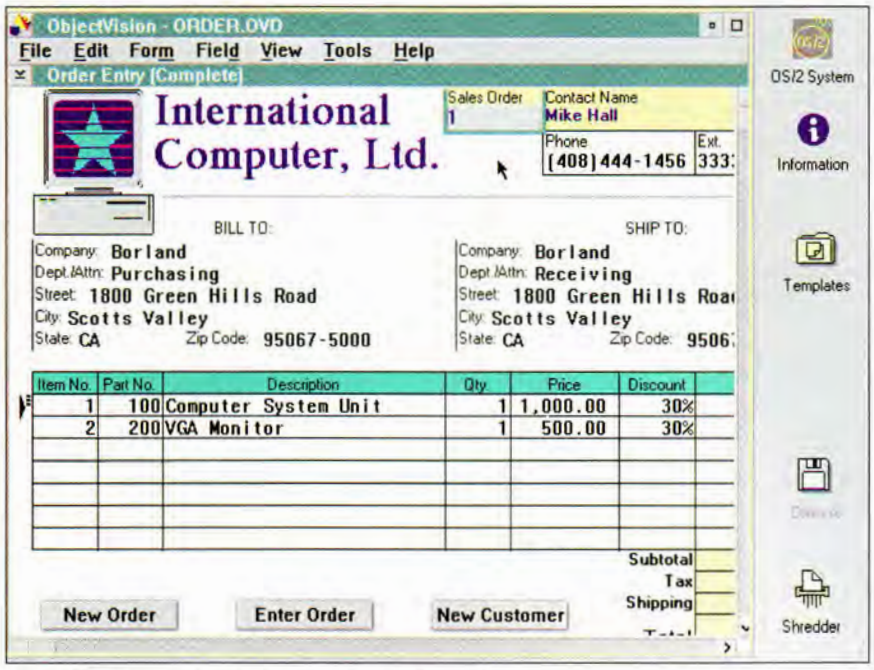
PM/FOCUS offers a rich graphical tool set complete with File Painter, Forms Painter, and Query Painter to build even the most complex applications without writing code. Application components such as databases, forms, and procedures, become simple graphic objects that can be manipulated and activated with a point, click, and drag of your mouse. The comprehensive array of list boxes, check boxes, multiple fonts, radio buttons and other graphical images, offer an intuitive interface that's remarkably attractive and easy to use. PM/FOCUS comes with its own proven database engine. Optional interfaces let you create sensational GUI applications for most other popular SQL databases. And PM/FOCUS is EDA/SQL enabled. This means you can have access to over 50 relational and non-relational file types on more than 35 different operating platforms.

PM/FOCUS. It could well be the reason why OS/2 was invented. Call your local Information Builders branch office or Micro Products Telesales at **1-800-969-INFO.** FAX (212) 695-3247.

IBI **PM/FOCUS**
Information Builders, Inc.
1250 Broadway, New York, NY 10001

## VISUAL APPLICATION CREATION

Hall believes that ObjectVision represents a new category of software designed to be used by both programmers and users. "Our ObjectVision 1.0 target audience," he says, "was users. But programmers picked it up and wanted to use it. And that's what created the agenda for ObjectVision 2.0...adding extensions to make it more powerful for programmers while keeping our franchise with our users." With the new version, Borland introduced the Object Bar and the Property Inspector, examples of its Menus On Demand™.




*ObjectVision's user interface paradigm is a collection of forms.*

"The idea that the things you deal with in applications are objects, and that you can get at the properties of those objects with a mouse click, is a user perspective in object-oriented technology," says Hall. "You can see it beginning with ObjectVision, but you'll also see it in subsequent graphical user interface (GUI) products from Borland. Finding that Borland's GUI products were set up using the same paradigm as the OS/2 2.0 Workplace Shell™ confirmed this aspect of our design." Hall believes that there are three "ABC" elements that make up any application: "There's the application interface, the logic (which we call business rules), and the ability to connect the application to databases or other applications."

### *Application Interface*

In ObjectVision, the application interface for business is envisioned as a collection of forms. Objects relevant to each form are displayed at the top of the screen; a developer drags objects such as fields, text, tables, and drawings from the Object Bar and drops them into the application. To specialize an object, for example, by changing its color or typeface, the developer clicks the right mouse button to get into its properties. Developers may also assign default properties by backing up one level and putting the properties on the object bar.

### *Business Rules*

"Business rules is an example of an area where Borland has unique technology," says Hall. The company uses what it calls a "decision tree," a diagram of the logic. In a spreadsheet, equations or expressions are attatched to cells, in ObjectVision trees are attached to fields. Explains Hall, "With a tree, you can introduce branching, which says depending on the values of other fields (the basis of the branching), I'm going to select different expressions. That's a value tree. The other category is an event tree, because in addition to calculations, you want certain actions to occur." In an event tree, actions are triggered when the user clicks a button: "Depending on the value of specific fields, there are several different actions available...load a form, bring in or store a record from a database, or close the application."

### *Connecting the Application*

Built into the product are engines for Paradox, dBASE, and Btrieve™. In ObjectVision for OS/2, Borland added an additional interface for the Database Manager™. Because the engine is not built in, OS/2 Extended Services™ is required for the interface to work. Hall notes, "You can connect to existing tables in those formats or, if they don't already exist, you can create tables."

# How to migrate without learning a foreign language.

# PL/I Package/2

Now there's a desktop application development environment that speaks your language. Introducing IBM SAA® AD/Cycle™ PL/I Package/2—the easy way to develop or move traditional mainframe applications to desktop systems. With PL/I Package/2, PL/I application development can take place in a workstation environment exploiting the OS/2® 2.0 platform.

If you're an existing PL/I user, now you can relieve your mainframe of a significant portion of its heavy workload and reduce development and production costs by porting applications to the workstation for maintenance and rework activity. If you have a large library of PL/I code, you can protect your investment and continue to use existing PL/I programmers to develop new applications on the workstation. And whether you're a new or current user, now you can utilize productivity tools and features available on the OS/2 workstation such as the ability to develop stand-alone OS/2 and Presentation Manager® applications, as well as applications targeted for MVS or VM.

PL/I Package/2 takes full advantage of OS/2 32-bit architecture while still providing all the familiar features: machine independence, structured programming, multiple data types, built-in functions, a macro facility and a record I/O capability using the facilities of Language Environment™ for OS/2. It continues to be an easy-to-use, highly productive language that supports commercial, scientific, engineering and systems programming tasks. And now with OS/2 support, you can make your move to the desktop, without bringing along a translator or learning a new language.

To order copies of PL/I, a free evaluation copy of PL/I or free literature, or to speak to a PL/I developer, call **1 800 426-3346**, ext. STL2.

## Introducing PL/I Package/2 for OS/2

- *PL/I Package/2 moves mainframe applications to desktop systems.*
- *Reduces development costs.*
- *Provides familiar PL/I features.*

IBM, OS/2 and Presentation Manager are registered trademarks and AD/Cycle, SAA and Language Environment are trademarks of International Business Machines Corporation. © 1992 IBM Corp.

IBM

According to Kathleen Garvey, product manager for ObjectVision, ObjectVision can be used as a front end for existing databases. Garvey says, "We have some software vendors that have created third-party ObjectVision applications, which can be migrated to OS/2 since the object vision (.OVD) files are platform independent. You can create an application on one platform and use it on another; it doesn't matter...There are actually two methods of interoperability, data interoperability (accessing the same data from both Windows and OS/2) and the OVDs, the actual application." Platform independence is especially important to corporate accounts investing in OS/2, as they must connect users operating on OS/2 with those using older Windows systems.

*Kathleen Garvey*

The second major feature added for the OS/2 version of ObjectVision was the REXX interface. "If you're a developer who's familiar with REXX, ObjectVision for OS/2 is a good way to put a nice Presentation Manager™ front end on your application," says Turpin. "The neat thing about REXX," adds Hall, "is that it's designed for both programmers and nonprogrammers. It's also very powerful, in the sense that sometimes it's not exactly the REXX you're after, it's the things you can get to through REXX. For example, you can do commands in the environment through REXX. You can write a REXX program that has no REXX in it whatsoever, but contains instructions to some other program so you can use it as a macro. You can also refer back to ObjectVision. In fact, the default is that if


Group Dynamics

Award-winning workgroup dynamics, from N/Joy. The first multifunctional, object-oriented business software designed for OS/2. Supporting development of complex documents on your LAN. With easy sharing of all text, spreadsheet and graphic data — all updated automatically via point-and-click hot links. Call 1-800-392-7724 now for a limited-time offer too dynamic to miss.

N/JOY®

THE WORLD OF OBJECTS®

Vienna Software Publishing, 1398 SW 15th Street, Boca Raton, FL 33486, 1-800-392-7724

**Announcing TLIB™ 5.0!**

**Version Control for DOS & OS/2**

- **The experts loved TLIB 4:**

*"...amazingly fast... TLIB is a great system."* **PC Tech Journal**

*"TLIB has features and power to spare... TLIB is easy to use and the fastest of the reviewed packages."* **Computer Language**

*"I will not program without it."* **Uptime Magazine**

- **Now TLIB 5.0 adds:**

Automatic branching. Automatic version labeling across branches. Language-independent preprocessor, for "conditional compilation" in languages which do not support it (like dBase). N-way-tree version numbers. Branch and whole-library locking. OS/2 support.

- **Plus the features they loved in TLIB 4:**

Check-in/out locking. Branching, for parallel development. Keywords. Full binary file support (does not depend upon CRs in the file like other products). Wildcard and list-of-file support; can create lists by scanning source code for includes. Can merge (reconcile) multiple simultaneous changes and undo intermediate revisions. Network and WORM optical disk support. Mainframe-compatible delta generator for Pansophic, ADR, IBM, Sperry formats. Includes integrated PD MAKE by L. Dyer; also integrates with Opus™ MAKE, Slick™ MAKE, others.

MS-DOS **$139**, OS/2 (with MS-DOS) **$195** + shipping.
5 station network: MS-DOS $419, OS/2 $595. Call for other sizes.

*Burton Systems Software*

PO Box 4156, Cary, NC 27519 **(919) 233-8128**

REXX doesn't recognize the instruction, it goes back to the application for execution. So, we use features like that to do self-registration: write a program in REXX and there's no REXX logic in it, just the ObjectVision commands that are necessary to self-register the REXX procedures."

*Bill Turpin*

With ObjectVision for OS/2, Borland added threading support. Explains Bill Turpin, "We put all the lengthy computation stuff in a separate thread, so we can fire off that thread and, if something's taking a long time, at least you can switch to another application. One of the reasons we did that is once you start adding REXX scripts into the application, there could be a fair amount of computation time. One of our REXX samples does a loop 1,000 times; while it's doing that you can go do other stuff."

"There were two differences in developing for OS/2 2.0 vs. developing for Windows," continues Turpin. "The threading was a new thing for us—we'd never done any threading before. It posed some interesting technical and testing challenges. And because what we did was largely a port of our Windows product, we had to retrofit some of that stuff in. If we had started from the ground up, we would have built it with threading in mind and it would have been a lot cleaner. The second thing is that we started last September, and the tool set has continually changed. So that added another level of complexity to the development process, doing things parallel to operating system and tools development is always harder than waiting until the operating system and all the tools are ready."


# OPEN SHUTTER

## Screen Capture & Conversion For OS/2

**MORE FUNCTION, LESS PRICE....WHAT A CONCEPT!**

At One Up Corporation, abundance in function without abundance in price is paramount. Consequently, you'll find our screen capture application, Open Shutter, provides the flexibility you need to capture, modify, and output your desktop images at a price you'll want to photograph!

**CAPTURE YOUR IMAGES**

Our easy-to-use capture process allows you to select any rectangular area, window, or the entire desktop and capture with a single user-defined keystroke or mouse click.

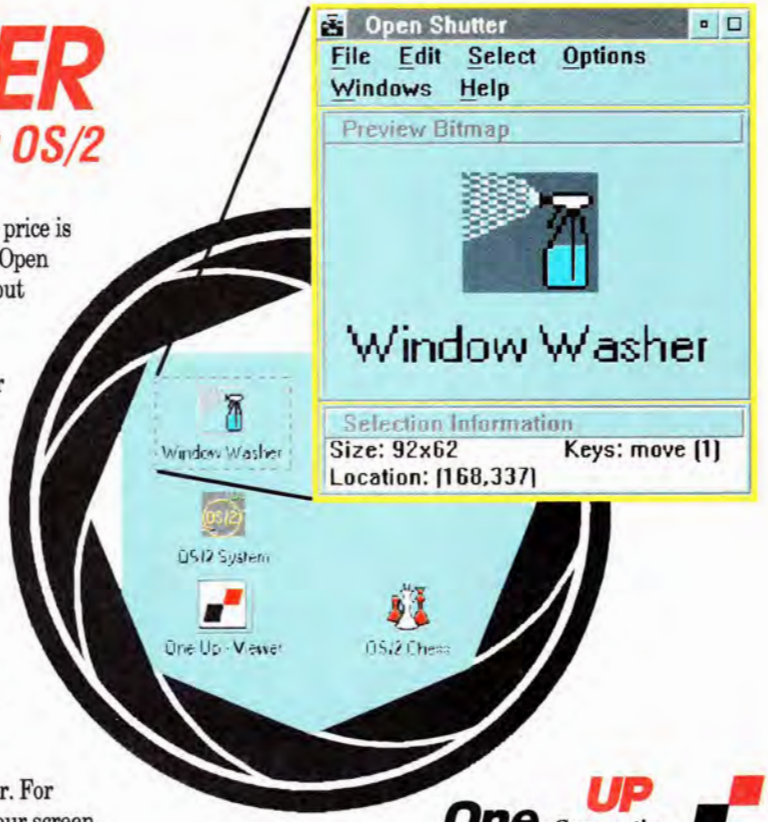
**MODIFICATIONS MADE SIMPLE**

Rotate your image at any angle, map your colors and modify your color palette, and stretch or compress your image to any dimension you need. Preview/compare multiple modified versions of your captured image prior to output.

**WHAT ABOUT OUTPUT?**

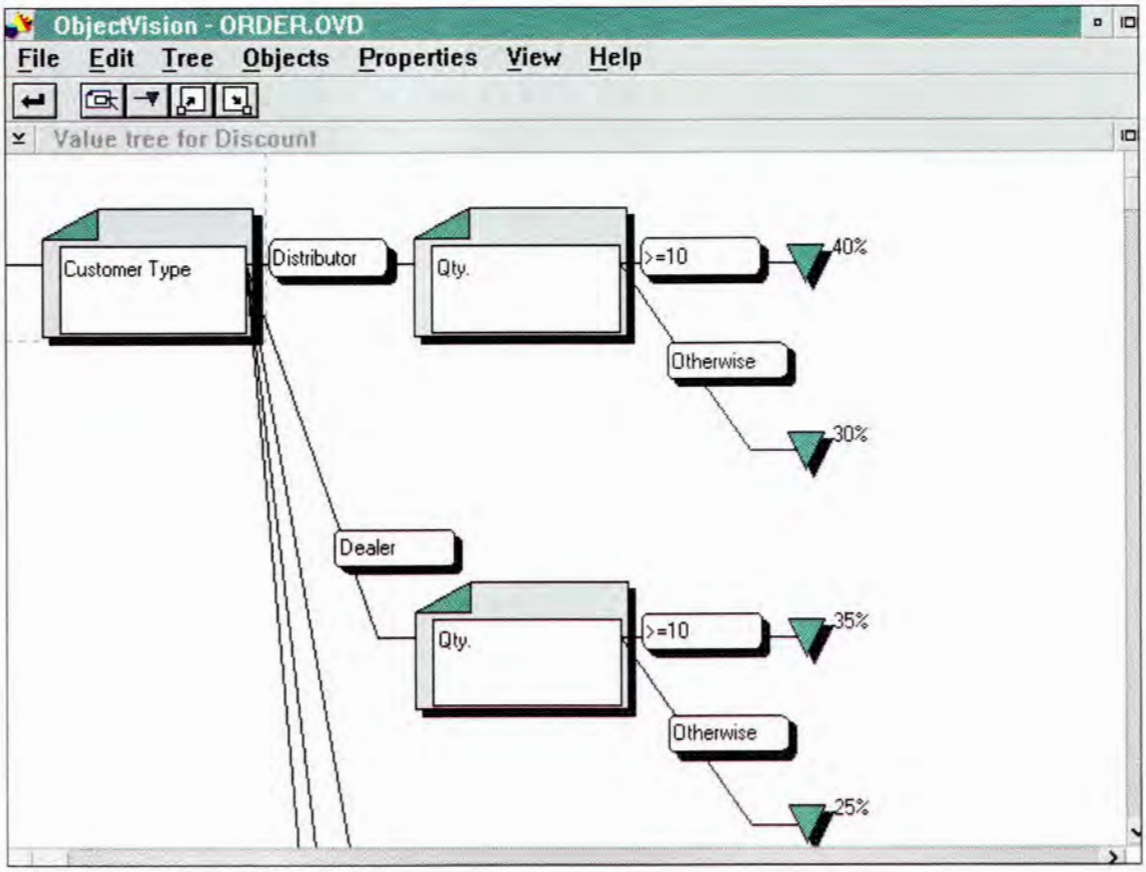
Open Shutter offers a varied selection of output devices including printer and clipboard. For soft copy, save your image as an OS/2 or Windows BMP, ICO, or metafile. Also, save as an OS/2 pointer, a Windows cursor, or PCX, TIFF, GIF, PICT formats and more!

So don't get discouraged by high price applications that don't deliver. For only **$59.95**, Open Shutter offers you a cost conscious solution for your screen capture requirements. Ask us about our competitive upgrade price, too.




**OS/2 is a registered trademark of IBM Corporation**
**Windows is a registered trademark of Microsoft Corporation**

One UP Corporation
1603 LBJ FREEWAY, SUITE 860
DALLAS, TEXAS 75234
1-800-678-01UP




*Value tree associated with an ObjectVision field*

ObjectVision's final version is compiled with the Borland compiler, and its .EXE file is 100K smaller than it was using the Microsoft compiler. The compiler is part of Borland's other planned OS/2 product, Borland C++ for OS/2.

## POWER APPLICATION CREATION

C++ for OS/2 will feature a 32-bit C and C++ compiler, a PM-based integrated development environment (IDE), a 32-bit version of Turbo Assembler™, and all the tools necessary to support OS/2 2.0 application development. "This will be a very complex product with a lot of pieces," explains John Ho. "We will provide a full set of development tools that range from a set of command-line tools like the compiler, linker, and assembler to a fully integrated environment that speeds up the edit-compile-debug cycle. We'll also have resource tools: you won't have to count pixels anymore; you'll be able to set up interactive editors." Borland is adding bitmap dialogue boxes, which generate resource files that define to binary, and a resource linker which allows programmers to do binding. They also plan to include a stand-alone graphical debugger to debug PM applications. This is a first for Borland; the current DOS and Windows versions use a character-mode debugger. These changes are helped by OS/2's multithread system.

*Stephen Drake*

Alex Lane notes that another feature of Borland C++ is the variety of precompiled headers. "That tends to save a tremendous

amount of time in GUI-type environments," he says. "With precompiled headers, you compile once and the compiler sends the information for subsequent compiles. You don't have to change the headers; you just cycle information straight in and it's comparatively very fast. I recall our Quattro Pro team saying that they were enjoying something like a 65% increase in compilation speed."

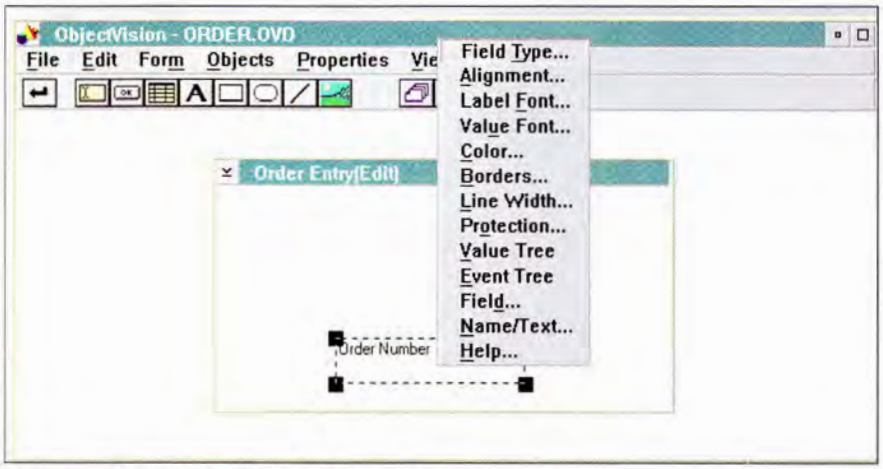
Ho explains that "Borland offers a one-stop solution because we believe that's the only way to impress professional users. Here's a set of professional tools that we built together, and that we know work together. We use our own compilers to build our tools. We turn on optimization by making sure our optimizer doesn't degrade the compilation performance. Having yourself as your own customer is a plus; we hear about bugs really quickly." Steve Drake, a member of the development team, adds that "the IDE is actually built in the IDE, and we've been using our command line tools practically from day one. So we're certainly a leading consumer of our own internal tools." He continues, "Our IDE mission is to provide tight integration of the Borland tools; however, we also provide a looser, Workframe/2-like integration of third-party tools." Borland is also providing integration of its command-line tools with WorkFrame/2.

Drake explains, "The IDE will use a few OS/2-specific features to enhance our functionality over the Windows and DOS versions. We're taking advantage of virtual memory to leverage space and multithreading capability to support background-building. The IDE allows the user to continue working in the foreground—for example, editing the next set of revisions of a project they're working on while the current revision is building in the background. We're also using the multitasking and particular pipes to facilitate external tool integration."

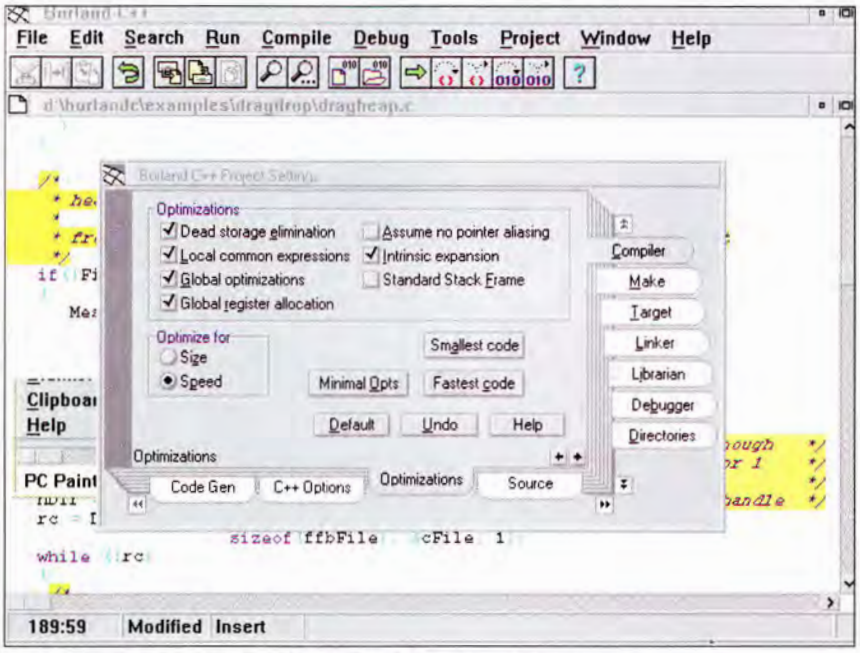
"The Borland C++ IDE for OS/2 2.0 shouldn't surprise anybody—it will look largely the same," says Lane. "Integrated debugging will be different but that's due to the nature of OS/2...Now the debugger has an actual operating system in which to operate."

## CUSTOMER-DRIVEN DEVELOPMENT

Borland is definitely a customer-driven business. "We can tell you everything about who our customers are. It's a way of doing business; it's a part of our corporate culture," says Brogan. The company is responsive to customer comments and tries to accommodate as many customer requests as possible. Technical support team members, who answer about 800 calls a day, keep developers

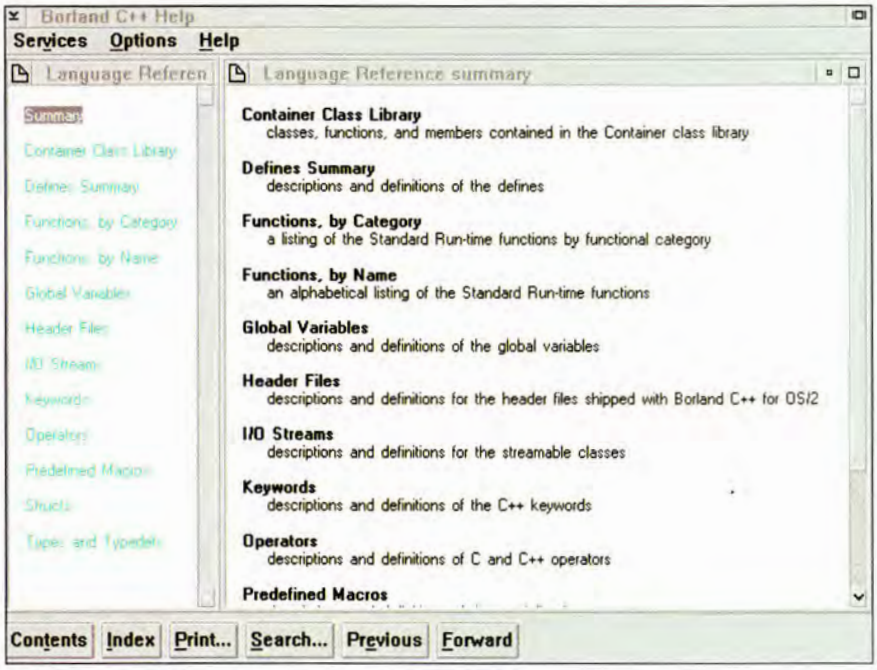



*ObjectVision's object bar and property inspector increase ease of use*

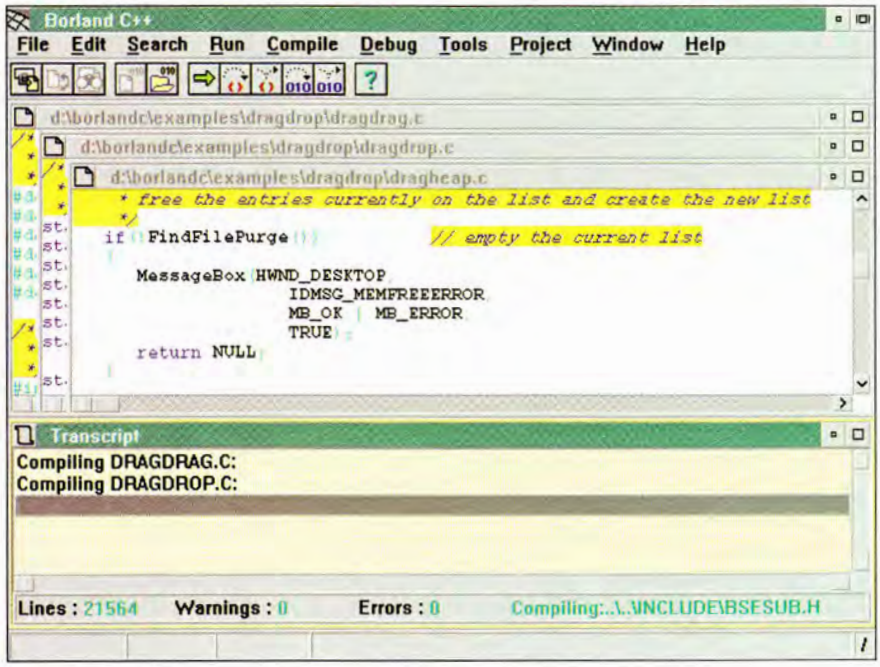



*Settings notebook for the Borland C++ integrated development environment*

informed as to customer concerns. They also give Borland an indication of new directions for its products. Explains Alex Lane, "They're the ones that come over and say 'Look, last week we got x number of calls on this.' Or a research and development person will come up and say 'Gee, I have this really neat feature, some new bells and whistles,' and we'll ask Technical Support how many calls we've gotten about this thing. If the product's been out and we haven't gotten call one on it yet, is it really that important?"




*Borland C++ help information*




*Transcript window and multiple edit windows in Borland C++*

Paul Gross adds that when it comes to product documentation, "the environment in which people are programming sort of sets the tone for what we have to do. Since there's good on-line help for OS/2, then we'd better provide the same. What we try to do is the right thing. We poll and find out what people like, what they read, what they throw away, and try to build useful documentation that can be used over and over."

## QUALITY COMES FIRST

Not believing in shipping "buggy" products, Borland doesn't announce ship dates, believing that the deadlines are less important than the quality of the product. This attitude paid off when the company won the 1991 J.D. Power and Associates Customer Satisfaction Survey from small, medium sized, and large businesses. Borland teams go on "bug-hunts" and form "bug councils," and utilize CompuServe. Matthew Stave, product group manager for Turbo Debugger and Tools, reads the IBMOS2 and OS2DEV forums daily.

Joy Lenz, quality assurance project manager for ObjectVision, described the quality assurance process. "We do testing to make sure the product is stable before it goes out to any customers...We write test plans and test suites (cases or scripts) for each product. There are at least three phases that we go through during the project cycle: alpha, beta, and gamma. The gamma phase is the 'dress rehearsal' before we ship. It means that there shouldn't be any known bugs, and we go through the manufacturing cycle to make sure [manufacturing is] ready, that all the parts are in. Normally, between the gamma version and what we actually ship there should be no code changes...If there are any, they've got to be what we call stop-ship bugs."

This committment to customers and quality is rewarded in the marketplace. In addition to the JD Powers recognition, ObjectVision for Windows received five awards in 1991. Readers of *InfoWorld* magazine voted it product of the year in the programming tools

# ONLY ZORTECH OFFERS C++ FOR BOTH NT AND OS/2 2.0—TODAY!

### *The Pioneers In 32-bit C++.*

Zortech has done it again. The pioneer of 32-bit C++ compilers for DOS in 1991 is back with the only high-performance 32-bit C++ compiler available today for both Windows NT and OS/2 2.0.

### *Meet The New Standard.*

Destined to become a new programming standard, the new Zortech C++ for Windows NT and for OS/2 2.0 fully support the AT&T CFRONT 3.0 standard C++, as well as ANSI C. That means complete support for precompiled headers and templates, as well as multithreading, double byte characters, and a C++ library for container classes, such as hash tables, vectors, lists, and more.

In fact, it's the only compiler that conforms to the IEEE-754 floating point standards and the NCEG 91-015 draft for numerical extensions.

### *The Best On Any Platform.*

No matter what operating systems you develop for, Zortech is the perfect choice. You can choose from 16-bit and 32-bit DOS, Microsoft Windows 3.x & Windows NT, IBM OS/2 2.0, SCO UNIX, and Macintosh.

### *C++ For NT And OS/2 Are Here. Where Are You?*

The new Zortech 32-bit C++ for NT and OS/2 2.0 is available now. Why wait? Get a jump on the future and on your workload.

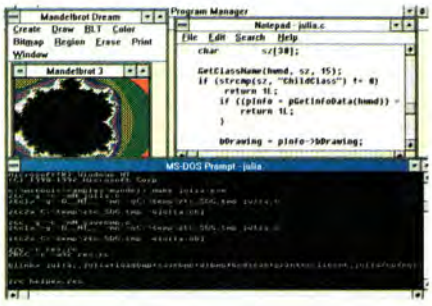
**Zortech C++ for OS/2 2.0** is just $499 ($249 for registered Zortech C++ users). To get additional information call **1-800-999-8846**. To order direct, please call: **1-800-228-4122 x 8352**.

For **Zortech C++ for Windows NT**, join the Zortech C++ Early Developers Program. See box below for full details.

You may also call **1-800-554-4403** for an instant FAX Data Sheet.

### *Join the Zortech C++ Early Developers Program.*

Get a head start on your Microsoft® Windows NT development by joining the Zortech C++ Early Developers Program. For only $499 ($249 for registered Zortech users) you'll get your own copy of Zortech C++, along with documentation that supports the Win32 Preliminary SDK. You'll also receive updates to support new pre-releases of Windows NT. Plus a final version of Zortech C++. To join, call 1-800-999-8846 or 1-408-252-1652. Registered Zortech C++ users should have their registration number handy.




© 1992 Symantec Corporation. *Price good in U.S. and Canada only — quoted in U.S. dollars. **Offer valid in U.S. and Canada only. International phone: Canada, 1-800-465-2266; Europe, 31-71-353111; Australia, 2-879-6577; others, 1-408-252-3570. All trademarks or registered trademarks are those of their respective holders.

SYMANTEC.

software and database front-end categories. The Software Publishers Association named it the "Best New Business Software." *BYTE* magazine gave it an Award of Merit and *Windows Magazine* honored it with its Annual Win Award.

*Matthew Stave*

No one knows what honors 1992 will bring to ObjectVision and Borland C++ for OS/2 2.0. But Borland isn't likely to be resting on its laurels. The company distributes corporate background information that gives an innovative view of a corporate culture: "Borland employees have been dubbed the 'barbarians.' Management and employees enthusiastically embrace the concept as a potent metaphor for Borland's way of doing business. The nomadic tribes of the steppes...were actually far more ethical and disciplined than the European civilizations they were confronting. They were austere and ambitious, eager for victory, but not given to celebrating it. They were organized around small collaborative groups that were far more flexible and fast-moving than the entrenched societies of the time." Borland brings this attitude to creating software, earning the company consistently high honors, satisfied customers and employees, and a swimming pool—a perk of which the original barbarians never dreamed.

*Joy Lenz*


***An Automated Regression Testing Tool for OS/2 PM***

# PMTESTER

o Test OS/2 PM and OS/2 Windowed applications
o Test Windows applications on OS/2 2.0 Workplace Shell
o Screen Capture, Comparison and Masking
o Real-time "Smart" playback of Test Cases
o Logged Test Case outcomes, start/end time and duration
o Batch execution of Test Groups and Test Cases
o Adjustable execution speed

And many many more features for only $595.00!!!
Runtime version also available for $200.00.

Other OS/2 PM products available for $50.00 with source:
**LISTWIN** - Desktop windows revealed
**SNAPSHOT** - Screen capture
**PMMINE** - Mine game

**Princeton Windowing Systems, Inc.**
**P.O. Box 7635, Princeton, NJ 08543-7635**
**(609) 921-6695**

## WHY WASTE VALUABLE TIME...

***WHEN YOU CAN GO STRAIGHT TO THE SOURCE?***

*Introducing the only magazine exclusively devoted to software developers working in the OS/2 environment.*

❒ **THE SOURCE for OS/2 tips and techniques, direct from Big Blue to you.**

❒ **THE SOURCE for those working with everything from LANs to multimedia to DBMS architecture.**

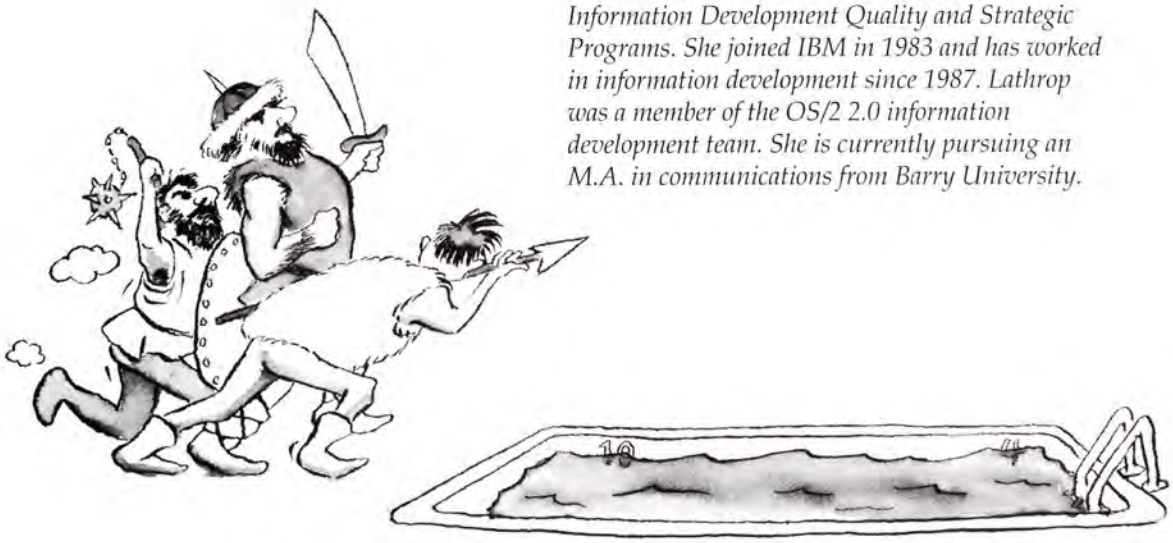
❒ **THE SOURCE for valuable news and information for OS/2 application developers.**

❒ **THE SOURCE for new software development ideas and products.**

*Subscribe now and pay just $39.95 for four big issues, delivered every single quarter to your office desk – straight from the source.*

SPEED YOUR ORDER!
1-800-WANT-Os2
1-708-647-5960
(OUTSIDE THE U.S.)
FAX: 1-708-647-0537

**Lynn Lathrop**, *IBM Corp., 1000 N.W. 51st St., Boca Raton, Fla. 33429. Lathrop is a senior associate information developer working in Information Development Quality and Strategic Programs. She joined IBM in 1983 and has worked in information development since 1987. Lathrop was a member of the OS/2 2.0 information development team. She is currently pursuing an M.A. in communications from Barry University.*



**Find Code *Fast*** • **Change Code *Fast***
**Learn Code *Fast*** • **Navigate Code *Fast***

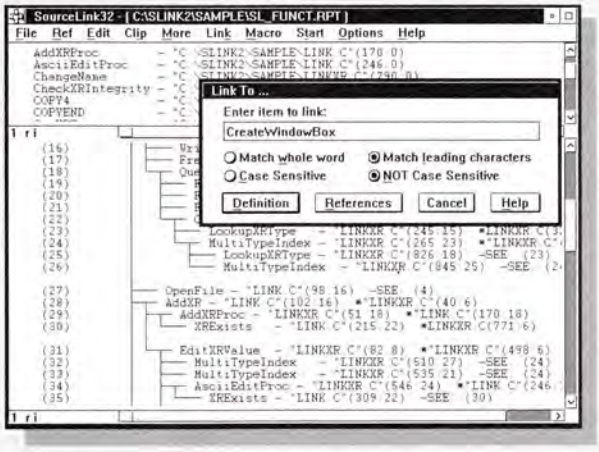
## *SourceLink*™

SourceLink is an OS/2 programming development tool combining the speed and power of hyper-link source code access with a fully functional editor. The edit screens are context sensitive. Click to access on-line help. Click-click to access your source code. With over 100 menu items to choose from, SourceLink is a feature rich programming environment.

Double click on a function, symbol or other string. Let SourceLink pop up a list of occurrences. Double click on each occurrence and SourceLink will immediately display the source code. Use the integrated editor to change or create new code.

Unmatched in speed and convenience for finding and changing source code. Ideal for porting code, changing code, analyzing code, and creating programs from existing code, templates and samples.

SourceLink will save you valuable programming time and energy! SourceLink is a refreshing new experience in programming!




***Features:***

- HyperLink source code access.
- Fully functional editor.
- Supports 'C', C++, MASM.
- REXX macro language interface.
- Presentation Manager GUI.
- WorkFrame/2 compatible.
- Generates function call tree displays.
- Context sensitive edit windows.
- Many file manipulation utilities.
- Find Files, Delete, Copy Files/ Directories.
- Context sensitive View Help.
- Multi-File/Dir. string search.
- Project configurable.
- Spawn compiles and commands.
- Create your own code templates.
- Multiple windows.
- Cross reference globals, symbols, dialogs, API calls, and more.
- Also available for OS/2 1.3.
- An OS/2 2.0 32 bit application.

***SourceLine Software, Inc.*** - *7770 Regents Rd #113-502 - San Diego, CA 92122 - (619) 587-4713*

***Your Satisfaction is Guaranteed***

Tools

# SourceLink: A Source Code Processor

*Bill Mueller*

*Ina Rath*

*by Bill Mueller and Ina Rath*

*SourceLink™ combines the speed of a hyperlink source-code browser with the functionality of an editor. It adds file-manipulation utilities to create a tool for learning, changing, porting, and creating code from templates, samples, and existing source code.*

Bob is using the traditional method for making changes to his source code. Let's watch him use his full screen programming tool.

"Let's see, now; exit to the OS/2 command line. Type `grep FillListBox( *.c`," says Bob. "Should I direct the output to a print file or just display it on the screen? I don't want to deal with the printer now, so I'll just display the results. Hurry...press Enter. Whoops, I forgot to use the `more` option and my output is scrolling off the screen. One more time. That's better. Now I can copy each filename and line number off the screen and make a find list for the search. With that completed, I'll type `EXIT` to get back to my full screen editor. Now, it's going to be really simple...Hotkey Alt-F for opening a filename...type `C:\MYAPPS\SOURCE\SLOWWORK\VERSION2\SLOWMAIN.C`. Once the file is displayed, press Alt-G and enter the line number from the search list...where's that piece of paper? Now I can change the argument to the first function call, and I only have 123 more to go."

Poor Bob. No wonder he hates making sweeping changes to his source code.

In the next office, Sally is using her new copy of SourceLink. Let's see how she's making those code changes.

Click-click; ZING! Click-click; WHAM! Click. Hot-key. Insert. "One more change completed," she smiles. *"I'll be finished soon."*

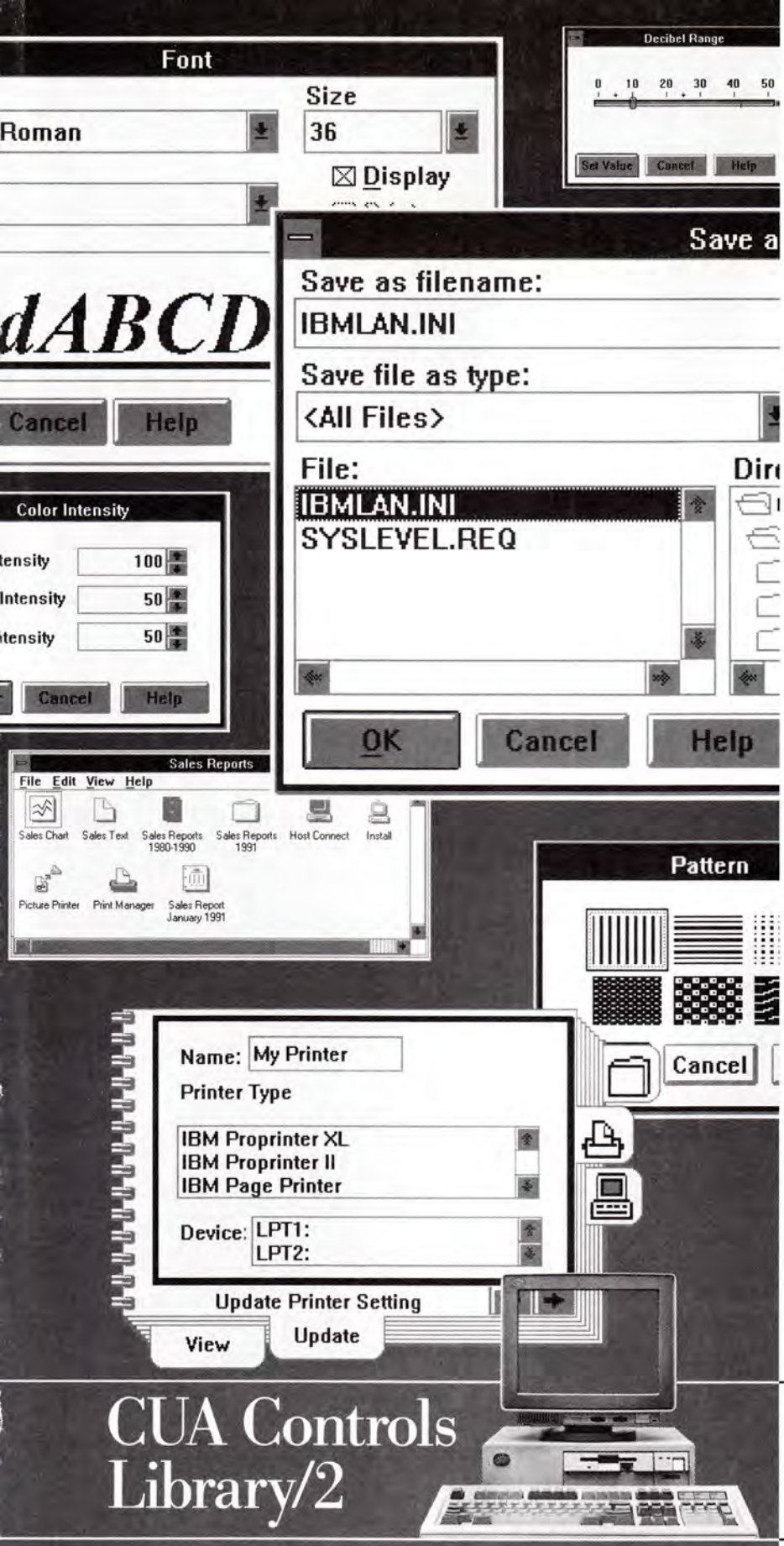
Let's rewind and watch her in slow motion.

Double click on the function name. Up pops a list of all occurrences of the function call (as shown in Figure 1). Double click on the first occurrence listed, and the source code is on the screen. Click on the argument name. Press F11 to select the word. Press F12 to insert the replacement. Done. Press Alt-B to get back to the search list and double click on the next occurrence.

## THE POWER OF SOURCELINK

SourceLink has been created for developers who need direct access to their source code. Performing a global string search with SourceLink involves no typing of filenames, no list making or list printing, no pondering over the file location of a function, and no guesswork.

SourceLink can be easily tailored to best suit specific programming needs. In the browse mode, it creates call trees, alphabetic function lists, lists of globals, ifdefines, symbols, and OS/2 API calls, and provides hyperlink access to any of these elements in the source code. You can access the definition or any reference of a source-code element that is part of the hyperlink.




# The computer company that never takes shortcuts is delighted to present seven.

They're on the screens you see here. And they can end the most boring part of your job—writing code for GUIs.

Introducing CUA Controls Library/2.™ It contains all the drudgery you'll never have to do again, if you're writing Windows™ or OS/2® applications on OS/2 1.3, Windows 3.0 or 3.1. It's all right here and it's all reusable. Over and over again.

With CUA Controls Library/2, you save time. You save money. And best of all, you can save your energy for the really creative parts of your work.

These screens are just a glimpse. From tables to graphics, from files to notebooks, from fonts to textures to tones, CUA Controls Library/2 does it all. To give you complete control over your program's functioning, organization, design and "look." All in a flash.

In a world as competitive as this, IBM definitely thinks you should take the easy way out. CUA Controls Library/2. For more information or to order, call **1 800 342-6672.**

# CUA Controls Library/2

- *Develop for OS/2 1.3 and Windows 3.0 and 3.1.*
- *Save time, money and resources.*
- *Call 1 800 342-6672 for more information or to order.*

IBM and OS/2 are registered trademarks and Controls Library/2 is a trademark of International Business Machines Corporation. Windows is a trademark of Microsoft Corporation. ©1992 IBM Corp.

IBM®

In the edit mode, SourceLink provides such features as multiple file, split-screen editing, multiple file string replacement, multiple undo and redo, automatic snapshot backup, cut, paste, copy, and delete. Code within parentheses or braces can be selected with a menu item. Developers can modify the macro creation of templates, comments, and function headings to fit personal coding styles.

The REXX macro interface, shown in Figure 2, gives access to the REXX macro language, plus over 100 SourceLink internal functions and some new REXX extensions for file I/O and callable PM entry, message, display, and list box panels.

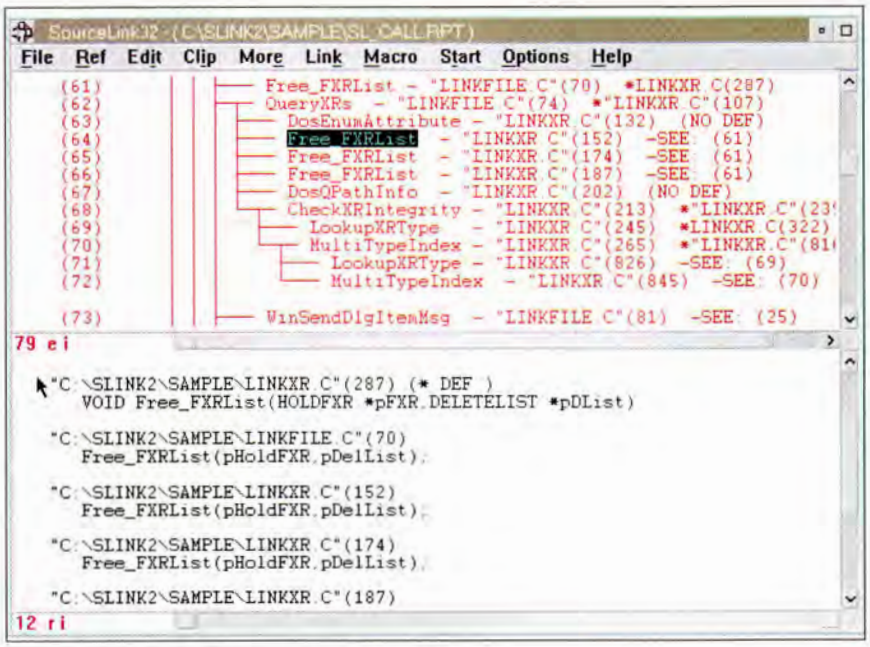



*Figure 1: Function call tree and hyperlink search list*

SourceLink also provides integrated file-manipulation utility functions. File and directory copy, move, and delete functions manipulate files. Drag-and-drop, compile error processing, and other features are fully compatible with the IBM Workset/2 development environment.

You can spawn a compile and have the error list pop up in SourceLink and click on an error line and immediately access the point of error. All SourceLink screens are hyperlink sensitive and editable. You can access IBM on-line help from within SourceLink by clicking on a keyword and pressing Help.

## FEATURES

This productivity product is one in a relatively new class of power tools that uses the versatility of the Presentation Manager GUI with the speed of native 32 bit OS/2 2.0 architecture. SourceLink is compatible with the IBM Workset/2 development environment and can be integrated into the IBM WorkFrame/2.

SourceLink contains over 100 menu items to access, manipulate, and change source code. Developers can choose the working paradigm that best fits specific requirements. Some find SourceLink helpful for learning new code, while others adopt it as a primary programming tool. SourceLink can be used with an editor or can work as a primary editor.

SourceLink functions in several categories:

- Source code navigation and documentation
- HyperLink source code access
- Source code editing
- REXX macro language interface
- View Help access
- File manipulation.

### *Source-code Navigation*

A byproduct of hyperlink database creation is a set of navigational aides and documentation display files from the source code. In addition to the function call tree and lists of globals, symbols, and user defined items, SourceLink provides a list of unreferenced functions which may be lying dormant in the code. These display files can be brought onto a SourceLink screen with a menu item or printed for permanent documentation.

### *HyperLink Source-code Access*

All SourceLink display screens are hyperlink sensitive for direct access to source code or the IBM on-line help facility, and global string searches or file finds. Clicking on an item in the list makes the source file appear. You can also create your own hyperlink sensitive project documentation.

### *Source-code Editing*

SourceLink provides multiple file, split-screen editing. A buffer ring gives immediate access to a previously displayed file. Files can be opened for edit or read only, displayed in a hex format, or viewed with word wrap. In addition to standard editor features, there are menu items to select code within braces or parentheses, perform multifile and multiple-string replacement, and so on. The undo-redo edit control shown in Figure 3 provides full flexibility in edit corrections.

### *REXX Macro Language Interface*

A full REXX macro language interface is used in SourceLink. In addition to REXX language functions, SourceLink provides over 100 internal function calls to access SourceLink routines with your own REXX macro programs. You can register a REXX macro file as a menu item or run an interactive session with full trace capability from within SourceLink.

The program provides REXX macros for automatic templating and comment formatting. They can be modified to suit personal coding styles, or you can create new macros. Input-output and Presentation Manager macro functions are also included to increase REXX performance and provide PM entry, display, and list panels from within REXX macro routines.

### *On-line Help Facility Interface*

SourceLink has a direct connection to the IBM on-line help facility. You can open a help file from a list of all your help files, browse through the help index, or click on a keyword to request help. Example code can be copied from the help files and pasted into source code. You can also size help windows and display your own code in SourceLink alongside the help facility.

### *File-manipulation Utilities*

SourceLink contains file manipulation utility functions including file and directory, copy, and delete. You can change file and directory file attributes, find files, spawn command file execution, and run macro files. SourceLink can copy selected files and subdirectories of a certain date to a set of disks, creating directories, and notifying you to change disks.

## PORTING CODE: A PRACTICAL EXAMPLE

We have just been given the task of porting a 16-bit OS/2 application to 32-bit. We will start by creating a `make` file for its source code with the IBM WorkFrame/2. To specify the files for hyperlink access, drag all the .C, .H, and .RC files from the WorkFrame/2 project into the SourceLink "List of Files" panel.

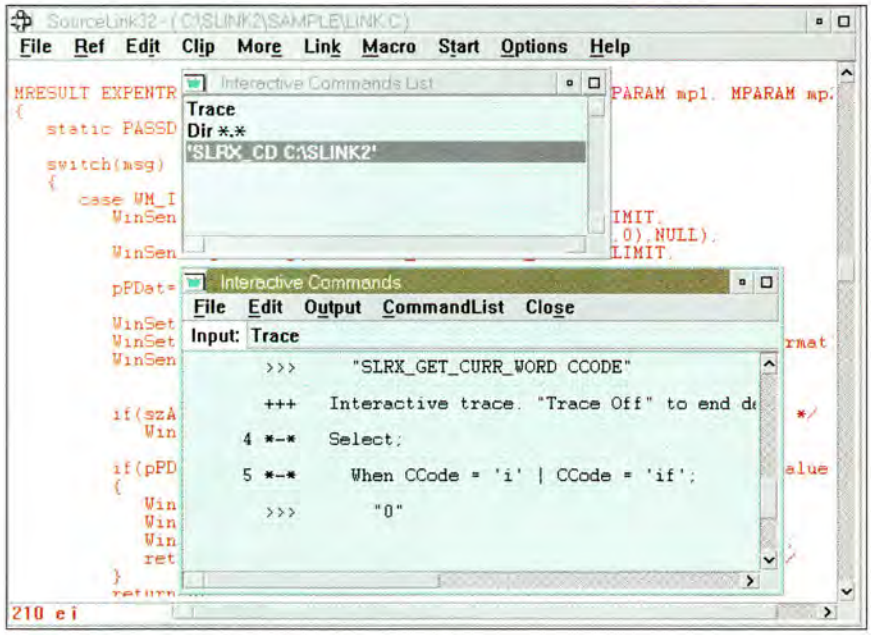



*Figure 2: Interactive REXX window within SourceLink*

*Figure 3: Multiple edit undo and redo panel*

The next step is to create the SourceLink hyperlink database by selecting "Build Db" under the "Link" menu. This analysis procedure allows you to specify the desired extent of hyperlinking. Specify all links including functions, symbols, globals, dialogues, and a wildcard list of OS/2 API

functions (`Win*`, `Dos*`, `GPI*`, and so on). This step creates a set of navigation and documentation display files, including a complete function call tree.

Now, take a look at the application to be ported. Click on the menu item to display the call tree. Because all display files are capable of hyperlink, double click on any node of the call tree to display the function's source definition. While looking through the source code of a function, you can double click on a called function to display the source code for this second-level function.

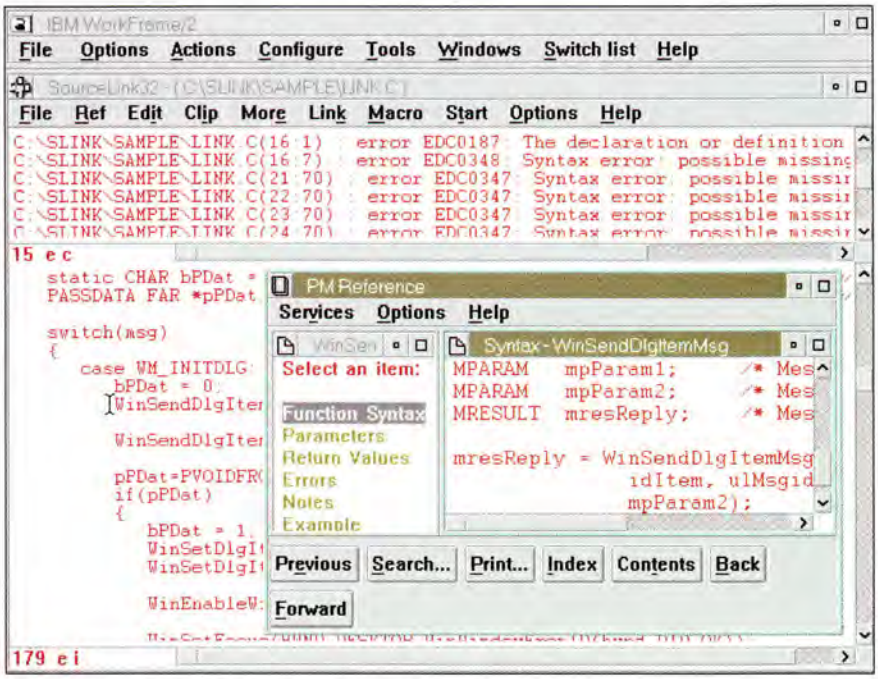



*Figure 4: SourceLink with WorkFrame/2, error list, and View Help*

*SourceLink eliminates typing, mistyping, and retyping filenames when accessing source code.*

Now let's find a function, global, or symbol, and double click with the right mouse button on the text. SourceLink will display all occurrences of this code element. Double clicking on any occurrence displays the source code at the point of reference. Hot key back to the list, and pick another occurrence of the item. This way, you can look at every reference of any hyperlinked item.

SourceLink eliminates typing, mistyping, and retyping filenames. There is no more list making following global searches and no accidently skipping over a referenced occurrence of a keyword. SourceLink helps you quickly find elements of your source code.

Compile the 16-bit code using the IBM C-Set/2 32-bit compiler. (The error list from compiling the first module in the `make` file is a reminder of how many subtle changes there are in the new 32-bit OS/2 API calls.) With the error list displayed in SourceLink, as shown in Figure 4, double click on any error line to display its source. Using the SourceLink hot key for help, pop up the on-line help for any API call to determine its proper argument types.

To find all erroneous API function calls, double click on the function in question. SourceLink displays a list of all occurrences of the function call. You can then access and change each function call in the source code.

Working through the compile error list in the first source file and correcting all occurrences through the remaining source modules helps you move quickly through the porting procedure.

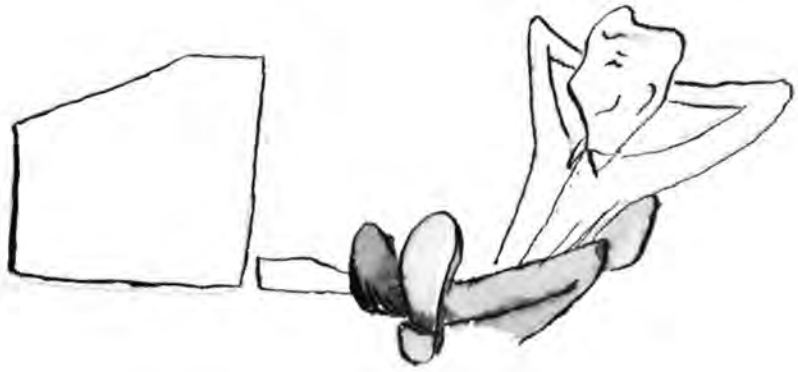
After making source code corrections for each module with the SourceLink editor, you can use either the IBM WorkFrame/2 or SourceLink to spawn a compile of the changed code and continue through the change cycle.

## SPEED AND POWER

SourceLink can access code quickly and powerfully and give you a new perspective on source code. Direct access gives you freedom from having to remember where code elements are located. If you are learning, analyzing, or changing code, SourceLink can save you time and pain. You will quickly find specific functional combinations that bring power and speed to program development.

**Bill Mueller,** *SourceLine Software Inc. 7770 Regents Rd., #113-502, San Diego, Calif. 92122, (619) 587-4713. Mueller is the founder and president of SourceLine Software and codeveloper of SourceLink. He has over 28 years of experience creating programming development tools and software applications. He can be reached on CompuServe at 72020,2713.*

**Ina Rath,** *SourceLine Software Inc. 7770 Regents Rd. #113-502, San Diego, Calif. 92122, (619) 587-4713. Rath is the codeveloper of SourceLink. She has over 20 years of experience designing and developing software on multiple platforms with an emphasis on low-level C routines and database functionality.*



*Direct access gives you freedom from having to remember where code elements are located.*

**Worried about testing your OS/2, networked applications? Not if you're using the Softbridge Automated Test Facility.**

If you're building OS/2 (or Windows) applications, you know that software testing is half the battle in the development cycle. If your applications are networked, they're even more complicated to test.

The Softbridge Automated Test Facility (ATF) was designed for unattended, centralized testing of OS/2 and Windows applications, networked or stand-alone.

ATF is used by corporations and software vendors who want to:

✓ **Build higher quality software**
✓ **Decrease development time & costs**
✓ **Make the impossible possible**

*Don't just take our word for it ...*

**Cooperative Solutions** used ATF to test Ellipse, and "Ellipse has been practically bug free." (*Datamation* - March 15, 1992)

"**Reuters Information Service** has used ATF for three months to test a client/server application. So far, ATF has enabled the firm to reduce the time required to perform 800 test cases from one month to one week." (*Network World* - June 22, 1992)

"**Bachman Information Systems** chose ATF because it's 70-80% better than anything on the market." (*PC Week* - June 15, 1992)

*If software quality is on your checklist, call 800-955-9190 and learn more about ATF.*

Softbridge

Softbridge, Inc. 125 CambridgePark Drive Cambridge, MA 02140 617-576-2257

Tools

# Smalltalking With the Database Manager

Theodore Shrader

*Combining the strengths of Smalltalk and DBM gives an application graphical and database power*

*by Theodore Shrader*

*As a development platform, OS/2 provides a robust environment for programmers who code applications for OS/2, DOS, and Windows systems. Several programming languages are available for OS/2 application development, including C, REXX, and Smalltalk. While C and REXX have their strengths, Digitalk's Smalltalk/V PM™ offers a language for rapid prototyping and the creation of graphical user interfaces for the OS/2 Presentation Manager interface. In addition, Smalltalk provides programmers with an object-oriented language that can be ported to multiple operating systems and graphical environments.*

OS/2 Extended Services (ES), released soon after the base OS/2 2.0 operating system, gives users the power of a relational database in Database Manager (DBM) and a strong communications component in Communications Manager. An application built with DBM can shield itself from the physical structure of data storage and can store and retrieve data through SQL. DBM also insures data integrity, provides referential integrity and concurrency, implements a transaction manager, allows roll-forward recovery, and supports OS/2 clients and servers as well as DOS and Windows clients. ES also comes with the DBM Command Line Interface (CLI) and graphical database front ends consisting of Query Manager and the Database Administration Tools.

Combining the strengths of Smalltalk and DBM gives an application the graphical and database power it can readily exploit in the OS/2 environment. Digitalk has provided a convenient way for Smalltalk programmers to access a relational database, giving access to most, but not all, database functions. For those who also program in other languages, such as C, Smalltalk programs can be extended to include calls to a dynamic link library (DLL) or to a background program written in another language. These background programs can in turn call the appropriate application program interfaces (APIs) to the database. For DBM, the APIs can initiate such actions as roll forward recovery and the import and export of table data. Static or dynamic SQL statements embedded in the program can, among other options, insert rows and query tables into the database. Alternately, the background program or DLL could call other database interface programs and share information. This article discusses both approaches.

## USING A RELATIONAL DATABASE

Why would a developer want to use a relational database in the first place? Wouldn't an ASCII file store information just as well? While it is true that storing data into an ASCII file and retrieving it as needed is faster than connecting to a database and querying a table, this process can quickly become unmanageable as the data grows in size and complexity. With an ASCII file, Smalltalk would have to parse the information being read and store it in structures, while preserving the structures and row order (for example, as an ordered collection of values). The same process must be followed when accessing a relational database, except the program does not need to physically parse the information; that is done by the SQL. To query only a subset of rows, the `WHERE` clause of the `SQL SELECT` statement can be modified to prune unnecessary rows from the returned report.

IBM®

# Ultimedia Developers Program

*An IBM Ultimedia service*

## Discover How IBM Can Help You Build Your Business

If you are developing multimedia solutions, we can support your creativity through the IBM Ultimedia Developers Program. The program provides your business with:

- Technical support
- Electronic communications
- Loaner Equipment
- Savings on IBM Multimedia Hardware and Software
- Discounts on Education/Technical Conference
- Multimedia news, trends, and information
- Business and promotional opportunities
- .... and more!

To request an information package or an application to join the IBM Ultimedia Developers Program, please call the IBM Multimedia Information Center at 1 (800) 426-9402, ext. D42.

ULTIMEDIA™

*Multimedia solutions from IBM*

Using a commercially available relational database also allows other applications to modify and query data through standard means such as SQL statements or existing graphical interfaces. It also makes the database structure public and makes it easier to modify and extend. Without a relational database, there is also the significant drawback that an application would need to maintain referential constraints on the data, a function already enforced by the primary and foreign keys in a relational database.

```
| dbmgr database table query1 query2 |

"Start the Database Manager, logon, and connect to the SAMPLE
 database."

dbmgr := EEDatabaseManager open.
dbmgr start.
dbmgr login: 'USERID' password: 'PASSWORD'.
database := dbmgr openDatabase: 'SAMPLE'.

"Query the VALVERDE table using a SELECT statement and send the
 returned report to the Transcript window."

query1 := database executeSQL: 'SELECT * FROM VALVERDE'.
query1 printResultsOn: Transcript.

"Query the TEXAS view using an instance of SQLTable and send the
 returned report to the Transcript window."

table := database table: 'TEXAS'.
query2 := table allRows.
query2 printResultsOn: Transcript.

"Disconnect from the SAMPLE database and stop the Database Manager."

database close.
dbmgr close.
```

*Figure 1: Opening and querying a database with the Smalltalk/V PM relational database interface*

## DEVELOPING WITH THE DBM SMALLTALK INTERFACE

Digitalk offers the Smalltalk/V PM Relational Database Interface, which consists of extra classes for Smalltalk/V PM. These classes allow applications to access the OS/2 Extended Services DBM and the Microsoft SQL Server™ products. The advantage to this setup is that developers neither need to know the database APIs nor deal with programming in another language—everything takes place in Smalltalk.

Figure 1 shows sample Smalltalk/V PM code designed to access and retrieve data from previously defined table and view objects in the `SAMPLE` database. Creating other objects and inserting, deleting, and modifying rows are done in a similar fashion.

With the Smalltalk/V PM Relational Database Interface, developers can access data in a table or view object in two ways. The first is the `executeSQL:` method, which executes a valid SQL statement. (These statements have similar syntax to those executed through the DBM CLI or as static or dynamic SQL in an application program.) Using this method makes SQL statements easy to recognize and change; however, the implementation of SQL can vary slightly from one database product to another and thus require unique SQL statements for each.

The second route uses `SQLTable`, whose instance and class methods are used to create, query, and modify rows in a table. This route hides the specific syntax of the SQL language from the program, allowing the program to be more easily ported between supported database products. However, for those familiar with SQL , this method may prove cumbersome to code and maintain. Although the first route may be slightly faster on slower machines, the performance difference between the two are minor.

## USING THE DBM COMMAND-LINE INTERFACE WITH SMALLTALK

Another way to access a database is to start and maintain a background session to the DBM. This route does not require the Smalltalk/V PM Relational Database Interface, and it allows developers to work in other languages that may be easier or more convenient for them. To access the database using this strategy, create a background server program that can be kicked off by the

**Reprints available from**
***OS/2 DEVELOPER***

Full or partial reprints are available for all articles appearing in OS/2 DEVELOPER.

Reprints can be customized to your specific needs. You choose color, size, layout, graphics, quantity, etc.

Use your customized reprints for :
***customer support**
***sales tools**
***marketing support**
***education**

*Call or fax today for a custom quote:*

**Andrea Varni**
**OS/2 DEVELOPER**
**Phone: 415-905-2552**
**Fax: 415-905-2236**

*World's most powerful tools for OS/2®*

**Hamilton C shell™**

**The superior alternative to the standard OS/2 command processor.** Faithfully recreates and updates the entire UNIX® C shell language. Created explicitly for OS/2 using modern incremental compiler technology. Not one line ported from or created on anything but OS/2. Very high performance. Extensive support for multi-threading.

**Features:** Full-screen command line editing • Filename and command completion • History • Arrow and function keys • Unlimited size command lines • Recursive filename wildcarding • Fully nestable control structures • Command substitution • Aliases and shell procedures • PATH hashing • Background threads and processes.

**Over 130 commands:** alias, cat, chmod, cls, cp, cut, diff, dirs, dskread, dskwrite, du, eval, fgrep, grep, hashstat, head, history, label, ls, kill, markexe, more, mv, popd, printf, ps, pushd, pwd, rm, sed, sleep, split, strings, tabs, tail, tar, tee, time, touch, tr, uniq, vol, wait, wc, whereis, xd and others.

**Meticulously adheres to OS/2 conventions.** Supports HPFS, long filenames and all networks under OS/2 1.2 or later and 32-bit and VDM applications under OS/2 2.0.

**$350.00. Unconditional satisfaction guarantee.** ($365 in Canada, $395 elsewhere.)

**Hamilton Laboratories**
13 Old Farm Road, Wayland, MA 01778–3117
Phone 508-358-5715 • FAX 508-358-1113

**Cover your application development needs with Across the Boards.®**

Across the Boards is the high-level PC-Mainframe interface that handles your connectivity needs, allowing you to develop applications quickly and easily. Across the Boards handles the complex details of communicating through:

- All major 3278/9 emulator cards
- SDLC/Bisync remote emulator cards
- Protocol converters
- LAN gateways

Plus, all are accessible through a single Application Programming interface.

With Across the Boards, you can access individual records, send information back and forth between programs, construct cooperative applications, and much more. All of which helps you cut development time and costs.

**PC Support**: DOS, Windows & OS/2
**Mainframe Support**: CICS VSE, CICS MVS, TSO, CMS & VTAM

**FREE 30-DAY TRIAL!**

**cfSOFTWARE**
**800-366-8756**
2454 East Dempster, Suite 201
Des Plaines, IL 60016
708-824-7180

Smalltalk program. The server program establishes and maintains its connection to a database until it is terminated by the Smalltalk application or until the application issues a "stop using database" command.

A simple way to access a database is to write a server program that accepts calls for the DBM CLI program, executes them, and returns the results to the calling application. Communication is handled by dynamic data exchange (DDE). The server program has several advantages:

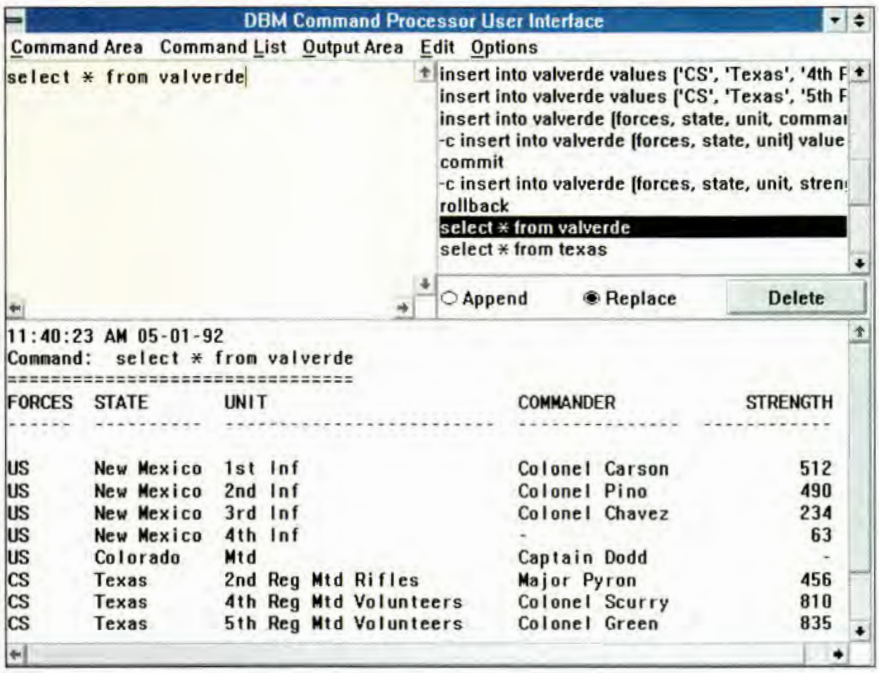



*Figure 2: DBM command line user interface*

- It can easily establish and maintain the database connection.
- It does not need to establish a special protocol for communication (DDE is used).
- It does not need to create specialized code to call database APIs, as it uses predefined statements to initiate actions (`DBM START USING DATABASE SAMPLE`, for example, is used to connect to the `SAMPLE` database).
- Its output is automatically formatted and ready for display (for example, query reports can be immediately posted in a text pane).

With such a scheme comes the performance overhead inherent in Smalltalk and the DBM CLI; large queries take time. The server program also needs to allocate memory dynamically and efficiently to accommodate reports from large queries. In addition, if the Smalltalk application needed the output in a different format than that returned by the DBM CLI, it would have to parse the returned report and format it appropriately. Since it uses the DBM CLI, the program is limited to DBM CLI supported commands (although these include essentially all available database functions). The server program's advantages—it can be quickly created and is reliable and easy to maintain—outweigh the disadvantage of performance degradation.

One possible sample program built with the server program approach is the Database Manager Command Line User Interface (DBMUI), shown in Figure 2. The interface is composed of three "panes." At the top left is the Command pane, where users can enter a DBM CLI command to be executed (without the "DBM" prefix). These commands can be stored in and retrieved from the adjacent Command List pane. Selecting a list entry either appends the chosen command to the contents of the Command pane or replaces its contents with the selected entry, depending on which radio button, Append or Replace, is selected. The radio buttons and a Delete button to remove list entries are located under the Command List pane. The Command List can also be executed in its entirety.

The Output pane is the last and largest text pane. When a command is executed, its output appears in this pane, along with header information such as the date and time the command was executed. When the contents of a Command pane are saved in the Command List pane, its output is also saved; each command entry is paired with its output entry. The user can use the clipboard functions to copy, cut, and paste text in the Output and Command panes. The contents of all panes can be individually or collectively saved to a file or printed.

When the DBMUI program starts, it creates a background OS/2 session and starts the server program within the session. The server is not a database server; rather, it is a program that

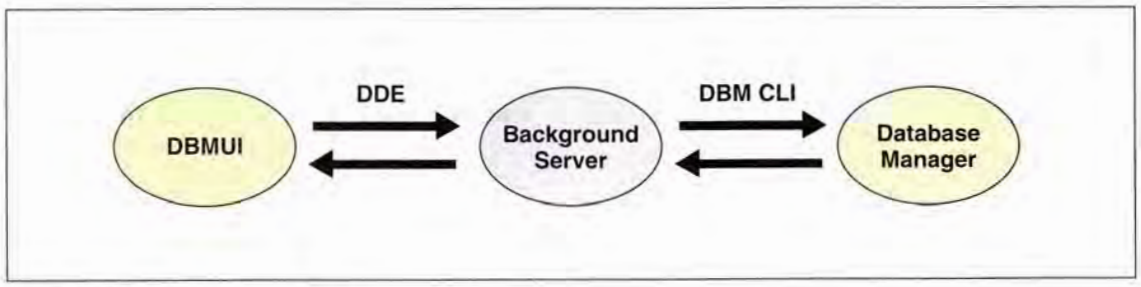



*Figure 3: DBMUI and background server communication*

accepts a command from the DBMUI application, executes the command by invoking the DBM CLI program, receives output from the CLI program, and, in turn, sends that output to the calling application. Figure 3 illustrates this series of steps.

The DBMUI program exchanges information with the server through DDE. (DDE protocols are publicly documented.) The server, written in C, responds to the PM messages `WM_DDE_INITIATE`, `WM_DDE_POKE`, `WM_DDE_REQUEST`, and `WM_DDE_EXECUTE`.

After the program receives the `WM_DDE_EXECUTE` message and retrieves the DDE data structure, the REXX I/O exit routine is registered, allowing output that would normally be returned to standard output (`stdout`) to be captured and saved in a buffer. Next, the program initializes the outpane buffer. This buffer, which contains the output report, will be sent back to the DBMUI program when the `WM_DDE_REQUEST` message is received and processed. Finally, after the program executes the DBM CLI command with the REXX SAA function, DDE acknowledgements are sent and the REXX I/O registration is dropped.

The Smalltalk interface code is much simpler than the code used to create the server program. After starting the background server program in a separate session, the DBMUI program establishes a DDE link and uses it to execute commands and retrieve their output. Figure 4 shows the method used for executing a command and posting its result.

A similar background server scheme could work in other languages as well. However, if the DBMUI program is written in a language other than Smalltalk, it may be faster to include the DBM CLI command call in the program itself, rather than use a server and incur a delay due to additional communication layers. Direct calls to the DBM APIs would also increase performance, but the program would then lose the benefits of the DBM CLI. In addition, the background server would have to be registered to the application to allow more than one DBMUI application to run on the desktop and keep messages from being sent to the wrong server.

## CONCLUSION

Which method is better for interfacing with the DBM? It depends on the application and programming environment. If a developer wants to write the program only in Smalltalk, the Smalltalk/V PM Relational Database Interface would be a better choice. However, if the application package consists of programs and DLLs written in more than one language, the background server approach would be more appropriate. Sometimes, too, developers cannot accomplish all the functions required for an application in Smalltalk/V PM, necessitating the latter choice.

The DBM program also supports DOS and Windows clients. Smalltalk code ported to these environments could access DLLs that allow the application to connect to the database server on an OS/2 machine to share information. (As the DBM CLI program runs only on OS/2, another interface must be created in the DOS and Windows environments.)

Smalltalk/V PM brings developers into a rich object-oriented programming environment that is well suited for graphical applications. Coupled with the DBM, Smalltalk/V PM programs can take full advantage of a proven relational database product.

*Smalltalk/V PM brings developers into a rich object-oriented programming environment well suited for graphical applications.*

```
cmdExecute
    "Execute command from the Command Pane and place its output
     in the Output Pane."
    | msgString cmdString |

    "*Get command and post header information in the Output Pane.*"
    cmdString := (commandPane contents) trimBlanks.
    self outputHeaderWithString: cmdString.

    "* Change mouse to busy pointer and disable interface. *"
    self busy.

    "* If background server exists, execute command and get output. *"
    "* nbackServer is an instance of the DynamicDataExchange class. *"
    backServer notNil ifTrue: [
        backServer executeCommandString: cmdString.
        msgString := backServer requestItem: 'Convoy' class: String].

    "* Post output in Output Pane and enable interface. *"
    outputPane appendMsgString: msgString; display.
    self notbusy.
```

*Figure 4: Smalltalk* `cmdExecute` *method*

## ACKNOWLEDGMENTS

The author would like to thank Futimoto Oshawa, Joel Dunn, and Phil Clem for their time and assistance.

**Theodore Shrader**, *IBM Personal Systems Line of Business, 11400 Burnet Rd., Austin, Texas 78758. Shrader is a senior associate programmer who has worked on database tools for the Database Manager product. He joined IBM in June 1989 and has worked in graphical application development since then. He holds a B.S. in computer science from the University of Texas at El Paso. (Go Miners! Win the NCAA!)*

## REFERENCES

*Guide to Database Manager* (IBM Doc. S04G-1013; 04G1013)

*Procedures Language 2/REXX Reference* (IBM Doc. S01G-6268-00; P10G6268)

*Procedures Language 2/REXX User's Guide* (IBM Doc. S10G-6269-00; P10G6269)

*Structured Query Language (SQL) Reference,* (IBM Doc. S04G-1012; 04G1012)

# Customize your communications.

Quadron's development tools for OS/2 provide the flexibility you need for custom co-processor communications.

## Match your protocols.

Use IBM ARTIC intelligent co-processor cards when you need extra ports or want to off-load protocol processing from your host computer. Then use Quadron software for custom communications with proprietary protocols, async, BISYNC, HDLC, X.25, and LAP-B. And if you need more than one protocol, no problem—just run them together on one card—at the same time.

IBM and OS/2 are registered trademarks of IBM Corporation.

## Write & debug with ease.

You'll feel right at home with Quadron software. Just program in standard C, and use our messaging API to link OS/2 threads with tasks on the card. Then use our symbolic debugger and real-time analysis tools in OS/2 windows.

## Protect your investment.

Perhaps best of all, our software is portable. Run what you write on any ARTIC card, on either ISA or Micro Channel. And if you later switch operating systems or platforms, just recompile.

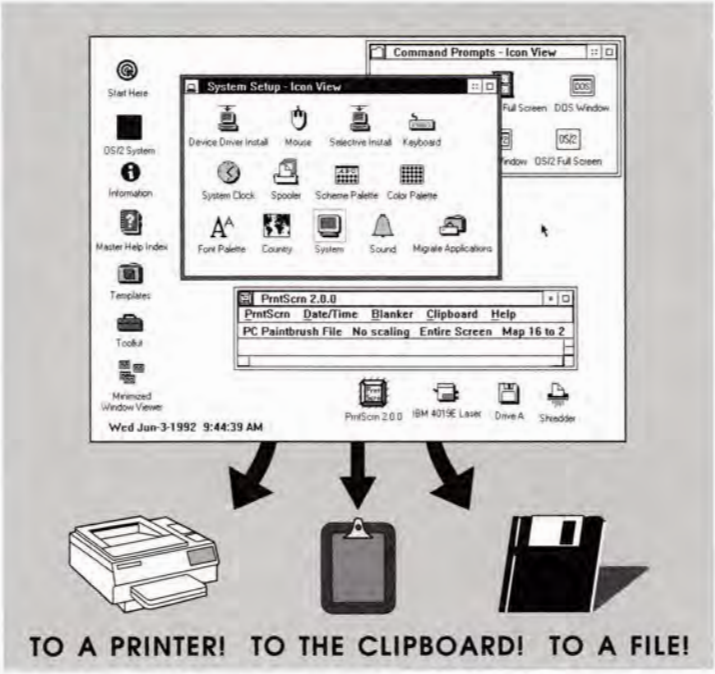
## Fit your applications.

Our easy-to-use software serves applications as diverse as telephone switching, financial transactions, satellite communications, and process control.

So call us. We've got an OS/2 communication solution—cards and software tools—ready to serve you today.

**Quadron**
Quadron Service Corporation
209 East Victoria Street
Santa Barbara, CA 93101
**805-966-6424**

# CAPTURE YOUR SCREEN




MITNOR Software
28411 E. 55th Street
Broken Arrow, OK 74014
Phone: (918) 357-1628
Fax: (918) 357-2869

Prnt Scrn™

**PrntScrn**™ is THE One-Step Screen Image Utility for copying a graphics image from the Workplace Shell screen to a LAN or locally attached printer, to the Clipboard, or to a disk file. The screen images appearing in this publication were captured and processed using the **PrntScrn** utility. The capture operation can be initiated from the keyboard (by pressing the print-screen key), from the mouse, or from another program. The captured image area can be the entire screen, current active window, current active client area, selected window, selected client area, or specified rectangular area. Disk file formats include PM Metafile, PM Bit Map, PCX, TIFF, GIF, and WPG. The "Color Mapping" feature provides the control needed to produce quality printed images from color screen images. **PrntScrn** has clipboard Viewing, Printing, Exporting, & Importing capabilities for Bit Maps, Metafiles, and Text files. **PrntScrn** includes a Screen Blanker and a digital Date & Time "information bar". **PrntScrn** is priced at $115 and is available from major software resellers, or contact MITNOR Software.

Tools

# Synectics: A New Distributed Processing Architecture

Steve Rosenberry

*by Steve Rosenberry*

*Synectics™ is a new distributed processing architecture that allows an application to locate and apply idle processing resources to its processing needs. This architecture provides the flexibility to adapt to changes in resource availability common in today's networks. By constantly adapting, Synectics provides the shortest possible application execution times given available processing resources.*

*Currently, client/server processing is the architecture of choice for distributed processing. Client/server architecture allows multiple users to share server resources. Unfortunately, the strength of client/server processing is also its weakness. Client/server centralizes the processing that accompanies client access, which eventually results in slow response times for all users.*

*Synectics supplements client/server architecture by providing a distributed processing architecture for applications that do not benefit from centralization. While it makes little sense to centralize spreadsheet processing, spell checking or software compiling, these applications could benefit from distributed processing.*

*Most LANs contain an immense pool of processing resources that can be tapped for distributed processing. Synectics manages these resources through task definitions: an application defines a task to be processed, and Synectics locates PCs with surplus processing resources for assistance. The task is then replicated, redefined, and distributed to the assisting PCs for processing.*

*Since all distribution decisions are made at run time, Synectics has the flexibility to use any currently available resources. If the application is executed on a stand-alone PC, only the only resources of that PC are used. In networked and multiprocessor environments where additional resources are available, Synectics applications leverage these resources for maximum performance.*

About the time OS/2 and LAN Server™ first became available, client/server processing came to networked PCs. Entire databases were no longer locked to all users because a single user was accessing a single record. Rather than simply sharing physical resources such as files and printers, servers shared processing with clients. This led to an increase in the use of LANs, which replaced mainframes in many situations.

There was only one snag: the most annoying flaw of mainframes, slow response times, also showed up in their smaller siblings, network servers. Servers now handle the database processing requested by all users. Too many requests and response times suffer.

Still, database applications benefit greatly from the central control point provided by the database server. But with other applications, such as word processors, spreadsheets, or CAD packages, concentrating processing on a single server only increases response time.

Synectics provides a distributed processing architecture for applications that work best on their own. With Synectics, applications can gain the power of distributed processing without the limitations of client/server technology.

## A COMPARISON OF TECHNOLOGIES

The term "distributed processing" is pervasive in the computer industry. What exactly does it

mean? Is client/server processing truly distributed processing? Is Synectics?

A brief look at the definition of distributed processing establishes a context for comparing the client/server and Synectics architectures. The client/server architecture may technically be called distributed processing, but it is better described as centralized processing. In contrast, Synectics truly distributes an application's processing among multiple CPUs.

### *Distributed Processing*

Distributed processing is a common enough term in today's computing vocabulary, but without context its use implies a number of very different technologies. In its broadest definition, distributed processing means that two or more processors are working together toward a common result. In the real world, it is more difficult to define. Distributed processing is a catchall phrase encompassing interprocess communications, client/server processing, distributed databases, and massively parallel computers.

### *Client/server Processing*

The most common distributed processing architecture used today is client/server processing. Client/server architecture consists of two processes, one executing on the client platform and another executing on the server platform. The client process provides a user interface that routes processing requests to the server process and presents the results to the user.

Client/server architecture has gained popularity through database engines. It provides a simple means of centralizing database management when the database must be accessed by a number of people. Figure 1 shows the client/server processing architecture.

The same centralization that makes client/server a natural for database engines hinders its use for a spreadsheet or word-processing engine. Concentrating processing at a server makes the server vulnerable to overload from too many clients. No matter how powerful the server, there comes a point when too many clients cause a slowdown in execution times. In the meantime, while the server is being overloaded the typical networked environment using client/server architecture wastes processing resources as PCs sit idly on the network.

With processing concentrated at the server, the client/server architecture also becomes vulnerable to single point failures. In its simplest form, client/server has no fault-tolerance: if the server fails, the client cannot process its data. Methods for overcoming this single failure point are available, but all require duplication of hardware and additional capital investment.

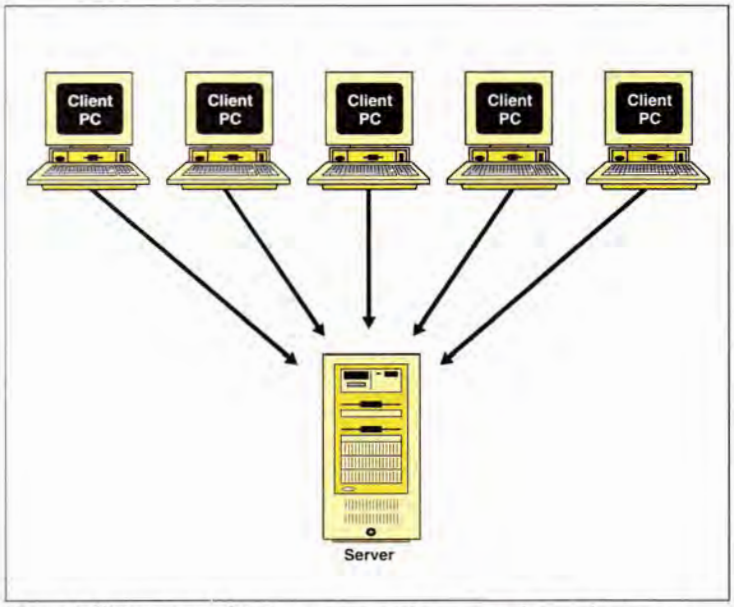



*Figure 1: Client-server architecture can overwork a server and cause slow response times for client PCs.*

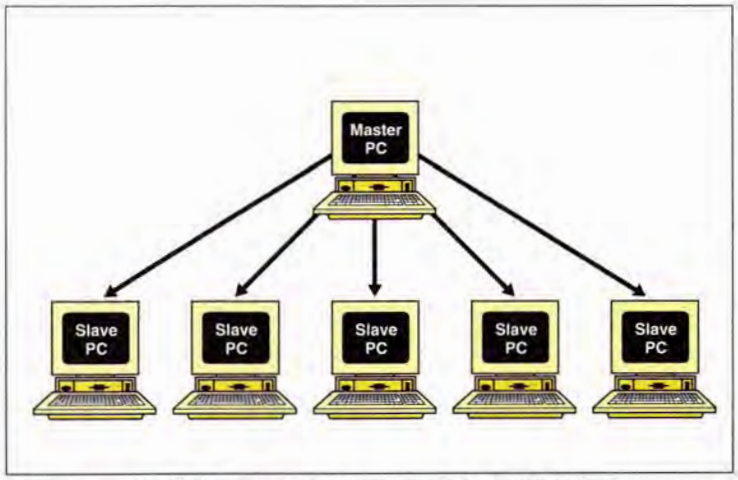



*Figure 2: Synectics divides processing between several slave PCs, reducing processing time.*

## SYNECTICS DESIGN GOALS

| | |
|---|---|
| **Fault-tolerance** | Eliminates single-point failures.<br>Exhibits high tolerance to hardware or network failures. |
| **Power-sensitive division algorithm** | Assigns the most work to the most powerful PCs while maintaining a balance with less powerful PCs. |
| **Noninterference** | Does not affect the use of a PC for other purposes. |
| **Low overhead** | Brings Synectics performance to problems taking one to two seconds. |
| **Resource coordination** | Provides network-wide coordination of processing resources for all Synectics applications. |

*Figure 3: Synectics design goals*

*Synectics provides the architecture to locate idle PCs and apply their resources to an application's processing needs.*

While client/server technology is technically distributed processing, its nature is not wholly distributive, as it tends to concentrate processing on a single PC.

### *Synectics Processing*

The new kid on the distributed processing block is Synectics. While client/server systems waste idle processing resources by concentrating processing on a single PC, Synectics provides the architecture to locate idle PCs and apply their resources to an application's processing needs.

In the Synectics architecture, an application creates and processes a task definition. The task definition provides information to Synectics about the task's processing requirements. When the task is processed, this information is used to determine an optimal method for its processing given the available resources. By replicating and redefining the original task definition, separate definitions can be distributed to different PCs for processing.

Synectics is less structured and more flexible than client/server architecture, in which all client requests must be handled by the corresponding server process. No single PC on the network is given the responsibility of processing a particular application's task definitions. Applications using Synectics can distribute their processing load to any PC on the network, and more important, processing can be distributed to more than one PC at a time. Figure 2 depicts the Synectics processing architecture.

Rather than executing a specific server portion of an application on a single PC, the Synectics architecture executes a generic task definition processing program on many PCs. Those PCs available when an application requests processing of a task definition may assist the application. With these "idle" PCs participating, the task definition can be processed in substantially less time than if processed on a single PC—particularly when the PC is an overburdened server already processing a number of other requests.

## SYNECTICS DESIGN GOALS

At the conception of Synectics, a number of design goals were specified as necessary for commercial success. These goals, shown in Figure 3, were deemed important enough that all other design decisions had to support them.

Fortunately, just as OS/2 provided the operating system for client/server processing, it provided an operating system with which Synectics could achieve its design goals. OS/2's priority structure and multiprocessing are used extensively to implement Synectics. Most of Synectics' design goals would be extremely difficult to meet in the single-tasking environment of DOS or even in the non-preemptive multitasking environment of

# OS/2 INDEPENDENT VENDORS

**Books/Magazines/Newsletters**
**Training Accessories**
**CBT Courseware**
**Audio/Video/Multimedia**
**Services/Consulting**

# DELIVER!

**Fax (914) 766-3788 for a FREE**
**OS/2 Independent Vendor Directory!**

I.V. League

OS/2® is a registered trademark of International Business Machines Corporation.

Microsoft Windows™. Even UNIX's™ current lack of multithreading would significantly complicate the development of Synectics.

### *Fault-Tolerance*

The highest priority design goal for Synectics was fault-tolerance. As an architecture becomes more complex, faults are more likely to appear. To gain commercial acceptance, the architecture must provide enough benefit to overcome the additional likelihood of failure or it must minimize the impact of any failure. Although the Synectics architecture is complicated, it provides the same level of fault-tolerance as single PC computing, and unlike the client/server architecture, Synectics has no single point of failure that would prevent all users from doing their work.

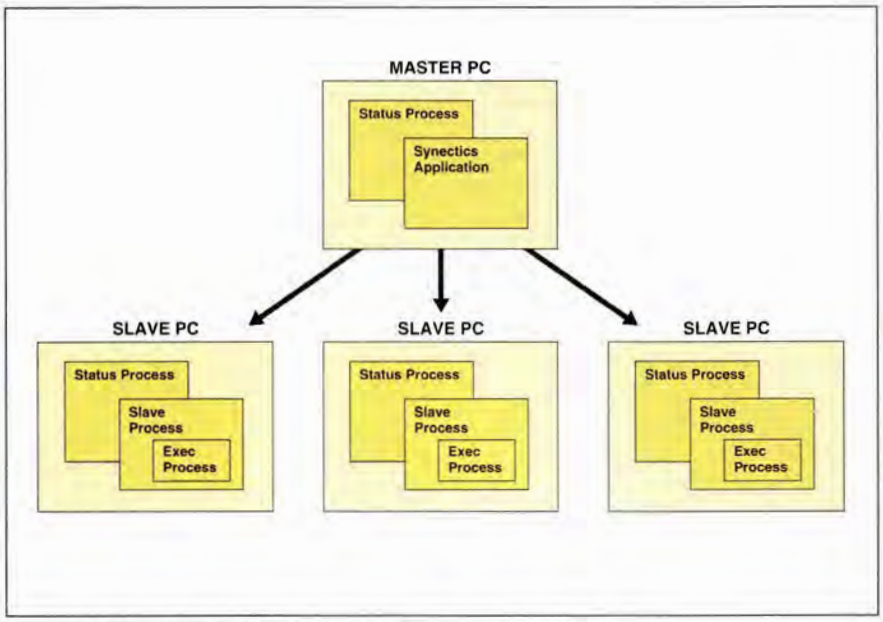



*Figure 4: Processes required for a PC to act as either a master or a slave*

A Synectics application executes using currently available processing resources. If a particular PC fails, another one may provide processing resources. In the event of a complete network failure and no available additional PCs, the application is processed locally.

Synectics applications must survive another source of failure that the client/server architecture typically avoids: PCs in use by a Synectics application may be turned off in the middle of processing. In contrast, servers for client/server applications are usually isolated from the general user and protected by uninterruptable power supplies.

Synectics applications may use any PC on the network. Because most assisting PCs are in use as desktop computers, they may be turned off at any time. Due to this likelihood of losing a PC's assistance, Synectics has built in fault-tolerance to recover from any hardware or network failure. Even if an application's processing is distributed across the network and the network fails, the application's processing will finish.

### *Power-Sensitive Division Algorithm*

When dividing an application's task definition among multiple PCs, the processing power of each PC must be taken into account. It makes no sense to allocate equal portions of the task definition to a 16MHz 386SX™ and a 50MHz 486™, as the 486 would finish in a fraction of the time required by the 386SX. Since the execution time of the 386SX limits the application's performance, shifting more of the task to the 486 would decrease overall execution time. A balance must be achieved so all PCs finish at the same time. Given the PCs involved, no better division of the task definition could be achieved.

Synectics uses a division algorithm that accounts for differences in PCs. The algorithm considers not only differences in raw power, but also the presence or absence of coprocessors. Synectics then redefines the replicated task definitions with the goal of having all definitions finish processing at the same time.

### *Noninterference*

To gain acceptance among users, Synectics has to be able to " borrow" their PCs without affecting their use. Users of PCs selected for processing a Synectics application should be unable to detect that their PC is or was being used (unless they want to know). A user must receive the same response times regardless of the use of his or her PC by a Synectics application. In addition, any files placed on the PC's disk drive should be removed. OS/2 handles the former through its priority

structure, while Synectics handles the latter by removing an application's files after they are no longer needed.

*Low Overhead*

By keeping overhead to a minimum, smaller problems may be divided among multiple PCs. This allows applications to not only improve performance for tasks with longer execution times, but also to improve response times to user input.

*Resource Coordination*

Without a standard architecture that can coordinate the use of idle processing resources by all applications, each application will implement its own scheme. Invariably, these schemes will conflict.

For example, each application might use its own process to measure the percentage of idle processor time. At worst, some processes would be entirely shut out, and the application would assume that those processors were fully utilized. At best, all of the applications would consider a processor to have less idle time than it does. In the end, no application would benefit because all would steal resources from each other rather than work together.

Synectics provides the necessary management of the processing resources for different applications to be distributed among different PCs. Synectics provides fair access to available processing resources for all applications, and applications can easily coexist on the network.

## SYNECTICS APPLICATION EXECUTION

From a user's viewpoint, Synectics has no affect on his or her interaction with an application. The application's user interface functions identically with or without Synectics. With Synectics, however, the application's performance is improved.

Users should, however, be aware of the additional processes required for Synectics applications to use the processing resources of networked PCs. These processes affect the role of a PC as either a master or a slave in the Synectics architecture. Figure 4 shows the process necessary to allow a PC to act as either a master or a slave.

*Master PC*

The master PC executes Synectics applications. Synectics places no limit on the number of master PCs that can be on a network. The only Synectics requirement for master PCs is the ability to share interprocess communications across the network.

*Slave PCs*

Slave PCs provide processing resources to master PCs. These resources are only provided if they are available. If a PC is being used, it will not provide processing resources.

For a PC to act as a slave, it must have access to the network and privileges to access a master PC's interprocess communications (IPC). When the slave does not have an actual user logon, it will be logged on with a user name and password specifically for Synectics. This account has IPC privileges only.

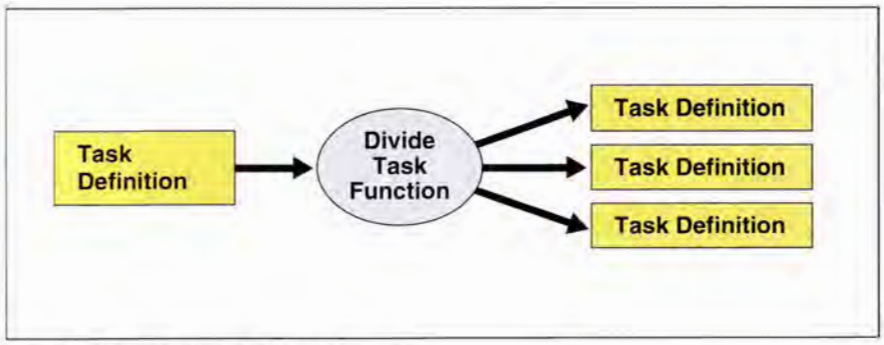



*Figure 5: The application's divide task function*

In addition to slave PCs accessed through the network, a master PC acts as its own slave. Although the mechanisms for using the master as a slave differ from using a networked PC as a slave, the two slaves are functionally equivalent. When it is necessary to differentiate between the two types of slaves, networked PCs acting as slaves are identified as remote slaves. The slave functionality associated with the master PC is the local slave.

### *Status Process*

The status process measures resource availability for a PC. These resources include a power rating, a percent CPU utilization, the operating system and version, and math coprocessor availability, among others. Both master and slave PCs must execute the status process to function. If a Synectics application is executed on a PC that is not executing the status process, the application will be processed locally.

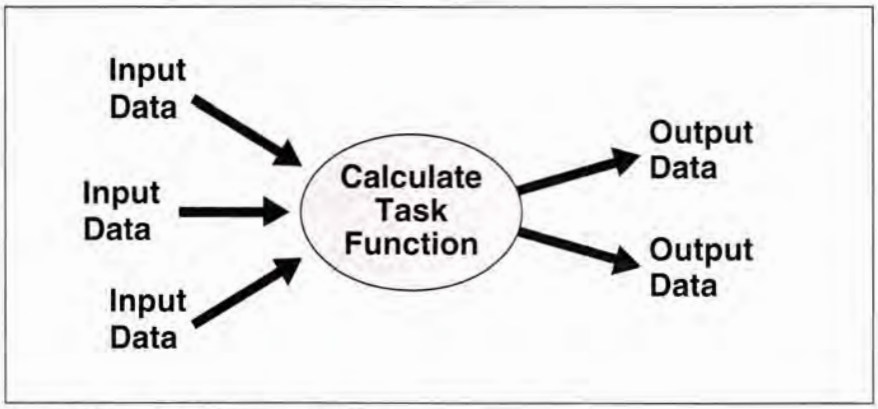



*Figure 6: The application's calculate task function*

### *Slave Process*

The slave process provides the slave PC's half of the communications interface between masters and slaves. It also handles the file caching for Synectics applications using the PC as a slave.

Mailslots and named pipes are used for communications. Initial requests for assistance, slave responses, and slave selections are communicated to and from the slave using mailslots.

Once a slave is selected for assistance, it connects to a named pipe created by the master. The slave receives its input data and executable files and returns the output data through the named pipe.

After processing a task definition for a master, executable files may be kept at the slave for future use. The slave process manages the cache of executable files for all masters, indexed by master and application. This allows each master to use different versions of the same application without conflict.

### *Exec Process*

The exec process is a child process of the slave process. Its sole responsibility is to perform the processing requested by a master PC and agreed to by the slave process.

By placing the actual application processing within a child process, the slave process is protected from faulty applications. Only the exec process will crash if the application fails catastrophically, while the slave process continues to run. This greatly enhances the continuity of the slave process and eliminates the need for another process to monitor the slave process. After the exec process completes its assigned processing, it signals its completion and then exits. The slave process may recognize either the completion signal or exit. If processing is successful, the resulting output data is returned to the master. Otherwise, an error code is returned.

### *Master Dynamic Link Library*

The master dynamic link library (DLL) provides the application interface to Synectics services and is used by Synectics applications as well as other Synectics processes. Its main purpose is to define an application's processing needs to Synectics and then manage available resources to accomplish the defined processing.

During processing, the master DLL will locate and select slave PCs for use, manage the division of the processing among the chosen slaves, communicate data and executables to the slaves, receive output data from the slaves, and manage the merging of the output data into its whole. Throughout, Synectics ensures fault-tolerance in the event a slave or the network fails.

### *Management Program*

The management program is an integrated installation and configuration program for Synectics. It can be used to install, remove, disable, and configure Synectics. It provides a number of configurable parameters used to optimize the performance of Synectics applications for a particular installation.

## SYNECTICS DEVELOPMENT BASICS

Developing a Synectics application is relatively simple. Anyone comfortable with basic OS/2 services will be comfortable using the Synectics services. The only additional concepts that developers must understand are the use of Synectics task definitions and the task functions provided by the application to manipulate them.

### *Task Definition*

Task definitions are the key to unlocking a network's processing potential through Synectics; they provide the information necessary for Synectics to manage the processing of an application's input data by slave PCs. A task definition includes information such as location and type of input data, names of executable files, file destination, and type of output data. It also provides information used to estimate execution times for task definition processing.

When an application's task definition is processed by slave PCs, the original definition is replicated and each copy redefined to create smaller task definitions. Each of these smaller task definitions is then assigned to a different slave PC for processing.

### *Task Functions*

If task definitions are the key to unlocking a network's potential, the application's task functions turn the key. An application supplies four functions used by Synectics to manipulate task definitions. They are the divide, calculate, merge, and abort functions. These functions, which reside in a DLL, are located by Synectics using ordinal numbers specified by the application.

The divide task function is called by Synectics when a smaller task definition is to be created for assignment to a slave PC, as in Figure 5. The divide function uses information supplied by Synectics to guide the creation of the new task definition.

*If task definitions are the key to unlocking a network's potential, the application's task functions turn the key.*


Gpf 2.0
Available Now
32 Bit Code Generation
CUA '91 Controls

# Let Gpf write the GUI you design

Using the powerful point and click visual programming environment of Gpf*, you can prototype, test and generate a complete OS/2 PM GUI in a few hours or days rather than the weeks or months required to hand code the same design. Even a relatively simple GUI can require writing thousands of lines of code, but with Gpf you simply draw your user interface on the screen. The integrated dialogue editor of Gpf permits actions and context sensitive help to be linked to controls as you create them. Gpf then generates error free ANSI C complete with embedded SQL statements.

Gpf is optimized to take full advantage of OS/2 PM, the most powerful and robust GUI system available. Since Gpf code directly accesses the PM API, there is no run time module to distribute with your application and no added overhead or royalties.

Gpf keeps the entire design definition in one file. This permits easy re-entry for updates as well as regeneration for different targets. 32 bit OS/2 and 16 bit OS/2 code generation.

Gpf SYSTEMS, INC.

Gpf supports:

- Simple and direct linkage of the interface to program logic, built in or user defined functions.
- Direct association of help screens with controls, and complete integration into the PM Help Presentation Facility.
- Flexible use of Presentation objects (fonts, colors, etc.) with controls and windows (client area and frame).
- Simple inclusion of bitmaps for use on About screens, user-defined buttons, and menu or pulldown entries.
- Automatic embedded SQL statements to read OS/2 DataBase Manager tables, directly into combo or list boxes.
- Multi-thread programming.
- Multiple source file generation.
- Automatic creation of controls that scale with window size.
- Inclusion of user defined controls

Try us out.

Order Gpf today for just $995.00
Call Gpf Systems Inc. at:
(203) 873-3300 or (800) 831-0017 - fax (203) 873-3302. Free demo software available
P.O. Box 414, 30 Falls Rd., Moodus, CT 06469

* GUI Programming Facility

The most important information supplied by Synectics is what percentage of the original task definition to assign to the new one. This percentage relates to the number of calculations to be performed while processing a task definition. If the original task definition requires 100 calculations to complete processing and the percentage for the new task is 30, the divide function should redefine the task so it will require 30 calculations. The remaining 70 calculations will be assigned to additional tasks as requested by Synectics.

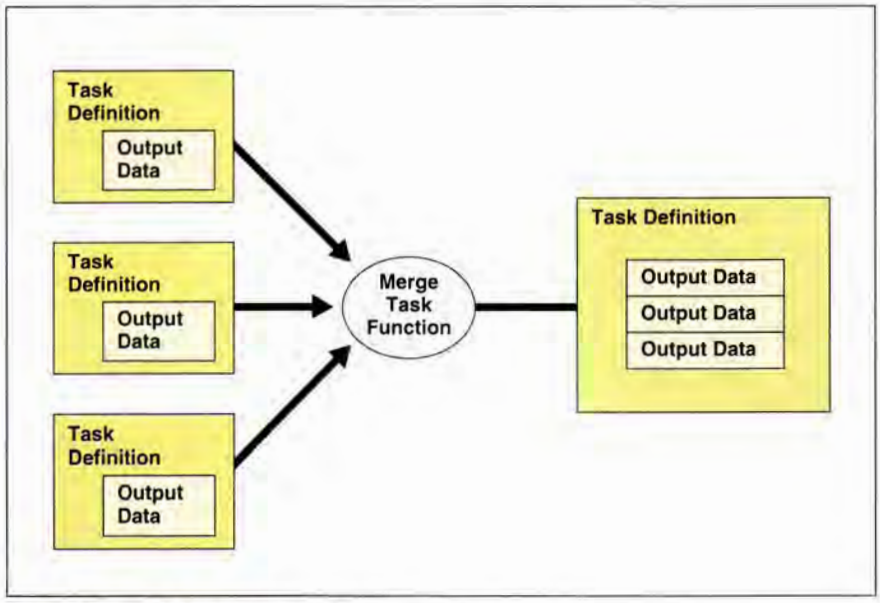



*Figure 7: The application's merge task function*

This percentage of the original definition is provided by Synectics as a guideline only. The application may adjust the percentage to meet its own requirements; Synectics then readjusts percentages for the remaining task definitions accordingly. Care should be taken when adjusting the percentage, as Synectics chooses values to minimize execution time. Modifying the size of a task definition could defeat the purpose of using multiple PCs.

The calculate task function is called by Synectics to process a task definition, an action that can occur on a master or slave PC. The calculate function, which manipulates the application's input data to generate the desired output data, as in Figure 6, is typically a modified version of the function that processed the application's input before the incorporation of Synectics.

Synectics calls the merge task function after the separate task definitions are processed and their resulting output data is returned to the master. The merge function consolidates the separate portions of output data to be presented to the user as a whole, as in Figure 7. If the original task definition is processed entirely by the master PC, the merge function is not used. In some instances, an application may not need a merge function even if the task definition was divided among multiple PCs.

The abort task function provides a clean method for halting the calculate function. It is used only on the master PC; the slave PC halts the calculate function simply by stopping the exec process. The abort function is optional; if it is not provided, the master PC will use an exec process to process the task definition. This ensures that the master can discontinue processing when necessary at the expense of additional overhead.

## SYNECTICS APPLICATION PROGRAMMING INTERFACE

The Synectics application programming interface (API) consists of four main categories: definition, processing, retrieval, and status. The API allows a developer to create and process task definitions and check on a task definition's status. Figure 8 summarizes the key Synectics API functions.

One of Synectics' design goals was to match the Synectics API as closely as possible to the OS/2 API. The same naming conventions, parameter types, and return codes were used where possible. In addition, task lists and task definitions are referenced through the familiar concept of handles.

### *Definition API*

The Synectics definition API creates and defines task definitions. To create a task definition, the application first opens a task list using the `PPCOpenTaskList` function. This creates a framework in which to process the task definition. Task lists will eventually allow multiple task definitions to be organized for sequential or parallel processing.

Once a task list has been obtained, a task definition is associated with the task list using

the `PPCOpenTaskDef` function, which allocates the system resources necessary to manage the task definition.

The remaining Synectics definition functions define the actual task. The `PPCSetTaskInput` and `PPCSetTaskOutput` functions define data locations and types (memory or disk based). These functions can be called multiple times for a single task definition if the data is maintained in separate locations.

Each call to the `PPCSetTaskExecutable` function declares an executable file required to process the input data. At least one executable, a dynamic link library containing the application's task functions, is required. Additional executables, including actual executable programs, may be defined for use by the calculate function as necessary. All defined executables are made available to slave PCs for use in processing the task definition.

The `PPCSetTaskCalculation` function provides information used to determine an appropriate processing strategy for a task definition. This information is used to estimate an execution time for the task definition on different PCs and to decide how many slave PCs to use. Although this information should be as accurate as possible, it is not critical. Given the variety of existing applications and PCs, Synectics adjusts for discrepancies in the calculation information.

*Processing API*

Once a task list and task definition have been created, they are processed using the Synectics processing API. There are two choices for processing a task list. `PPCProcessTaskList` processes the task list synchronously (returning only after processing is completed), while `PPCProcessTaskListAsync` processes it asynchronously (beginning the processing but returning immediately to the calling function).


# *Arcadia* Workplace Companion

**An OS/2 Personal Information Management System**

**Clock/Calendar**
Analog/Digital Clock
International Time
Julian/Weeks/Holidays
Launching Pad

**Appointment Book**
Daily/Weekly/Monthly/
Yearly Views

**Telephone/Address Book**
Dialer and Phone Log

**Call Management**
Contact List and
To-do List

**Arcadia Technologies Inc.**

735 West Duarte Road, Suite 207
Arcadia, CA 91007

**Tel: (818) 446-6945**
**Fax: (818) 447-4212**

OS/2, CUA, IBM are registered trademarks of International Business Machines, Corporation. Arcadia Workplace Companion is a trademark of Arcadia Technologies, Incorporated

| | |
|---|---|
| **Definition API** | PPCOpenTaskList( )<br>PPCOpenTaskDefinition (TaskList)<br>PPCSetTaskInputData (TaskDefinition, InputData)<br>PPCSetTaskOutputData (TaskDefinition, OutputData)<br>PPCSetTaskExecutable (TaskDefinition, ExecutableFile)<br>PPCSetTaskCalculation (TaskDefinition, CalculationInfo) |
| **Processing API** | PPCProcessTaskList (TaskList)<br>PPCProcessTaskListAsync (TaskList) |
| **Retrieval API** | PPCGetTaskInputData (TaskDefinition, *InputData)<br>PPCGetTaskOutputData (TaskDefinition, *OutputData) |
| **Status API** | PPCQTaskStatus (TaskDefinition)<br>PPCQSlaveInfo (TaskDefinition)<br>PPCQListInfo (TaskList) |

*Figure 8: Key Synectics API*

*Incorporating Synectics into an application is usually a straightforward process.*

`PPCProcessTaskListAsync` creates a new thread to process the task list, allowing the application to continue with other processing. Often, the application continues by providing processing status to the user as processing occurs in the new thread.

The previous functions search for slave PCs to assist in processing. If additional processing resources are located, the task definition is divided among them; if not, then task processing occurs locally.

### *Retrieval API*

The retrieval API returns information about a task definition. Most retrieval functions have a corresponding definition function. Typically, retrieval APIs are used to locate data during the application's divide, calculate, and merge functions.

### *Status API*

The status API provides feedback to the user during and after processing. Studies of human nature have shown that people are more likely to wait in line and will be more satisfied with the outcome if they know beforehand how long they can expect to wait. Because different executions of a Synectics application may require different amounts of time, feedback on estimated execution times satisfies a basic user need.

During processing, the `PPCQTaskStatus` function returns information on individual task definitions created by the application's divide function and assigned to separate slaves. Key information returned by `PPCQTaskStatus` is the estimated completion time for each slave. Additional information includes percentage completed, a return code if completed, and the status of the slaves processing the task definition.

For accurate estimates of completion time, the application's calculate function calls the `PPCSetTaskEstimate` function. `PPCSetTaskEstimate` requires only a percent completion value to estimate a new completion time for a task. This information is then provided to the application through the `PPCQTaskStatus` function.

The `PPCQSlaveInfo` function can be used once task list processing is completed. This function provides status information from the perspective of each slave PC used during processing, including how long it took to establish communications, transfer data and executable files, and calculate and return the output data. Other information indicates the slave's return code for processing the task definition and the final status of the slave. A slave's status indicates whether it was a local or remote slave and whether the task succeeded, failed, or was aborted.

## SYNECTICS APPLICATION DEVELOPMENT

Programming a Synectics application is much like programming any other OS/2 application. Due to the use of structured programming techniques by applications and the modularity of Synectics processing, incorporating Synectics into an application is usually a straightforward process. There are five basic steps to modify an application to use Synectics:

- Decide at what level in the application's function call tree to insert Synectics.
- Group as many input parameters in as few structures as possible and organize referenced data so it can be described easily with a pointer and size. Do the same for output data.
- Develop the algorithm for redefining a task definition to allow processing by multiple PCs.
- If necessary, develop the algorithm for merging the output data received from processing multiple task definitions.
- Incorporate the Synectics API to define and process task definitions.

## SYNECTICS' FUTURE

### *Improvements*

The first improvement scheduled for Synectics is to bring functionality to the task list. This will allow an application to place multiple task definitions on the task list for processing. The application will be able to specify the order in which task definitions are to be processed as well as any dependencies between them.

As experience is gained with Synectics, the algorithms for determining which slave PCs to use and what percentage of a task definition to assign to each chosen PC will be improved. The first improvement in algorithms will incorporate communication delays between PCs. Currently, Synectics assumes that communication times are insignificant compared to execution times. Even if this assumption is wrong, Synectics is designed to finish without penalizing the user.

A third enhancement will be the pre-installation of executable files on slave PCs. Although this is not required, it would be beneficial to applications requiring a number of executable files, or to those for which the executable files are large. It also makes sense in environments where all PCs are using the same application. Rather than duplicating an executable file by passing it to a slave PC that already has it, Synectics will use the executable file native to the slave PC.

### *Object-Oriented Programming*

Those familiar with object-oriented programming may recognize the natural tendency to treat task definitions as objects. Most Synectics data structures can easily be encapsulated as objects. Once this is done, an object-oriented version of the Synectics API can provide straightforward programming for Synectics applications.

*Most Synectics data structures can be easily encapsulated as objects.*

### *Multiprocessor Environments*

One of the most exciting prospects for Synectics applications is the advent of symmetric multiprocessing operating systems. These operating systems will provide the power of multiple processors to Synectics applications with minimal communication delays. These systems highlight Synectics' ability to maximize the processing potential of a single version of an application in any environment. For developers, this minimizes development costs in the long run by allowing Synectics to manage different environments and platforms. For users, this means less expensive software, maximum performance from the software in their particular environment, and the ability to economically add processing power as needed.

### *Network Environments*

Currently, Synectics works with IBM LAN Server and Microsoft LAN Manager™ networks. Support for Novell Netware™, IBM APPN™, and Banyan Vines™ will bring the power of Synectics to more applications and installations.

### *Mixed Platform Environments*

Finally, by incorporating the technology used for remote procedure calls between different platforms, a Synectics application will be able

to execute across different platforms. With feedback on the types of platforms currently available for processing, the application can maximize its performance by giving those portions best suited for mainframes to a mainframe or assigning floating point calculations to the strongest floating-point processor.

## CONCLUSION

While client/server architecture will continue to be the favored distributed processing architecture for database applications and others that benefit from a central point of control, Synectics provides an alternative architecture for other types of applications. Synectics is a distributed processing architecture that allows applications to execute at peak performance in all environments—networked or not. It provides built-in fault-tolerance to hardware and network failures and has the flexibility to adapt to a changing network environment by working with currently available processing resources. As the status of slave PCs changes from idle to busy, Synectics applications will use other slaves to satisfy their processing needs. At all times, Synectics attempts to provide applications with the highest performance possible given the available processing resources.

**Steven M. Rosenberry**, *Parallel PCs Inc., 1404 Durwood Dr., Reading, PA 19609 (215) 670-1710. Rosenberry is a founder and president of Parallel PCs Inc. He has been involved with OS/2 and the development of Synectics since OS/2 1.0 was in beta testing. He holds a B.S. in electrical engineering and an M.B.A., both from Lehigh University. Rosenberry may also be reached through CompuServe at 70413,1712.*

**"Professional programmers will be pleased with the meat in this book"**
*- Dr. Jim Gray in Database Programming & Design*

**Client/Server Programming with OS/2 2.0, 2nd Edition**
1152 Pages
*By Robert Orfali and Dan Harkey IBM Corporation*
ISBN 0-442-01219-5
IBM Order No. G325-0650 $39.95

# THE DEFINITIVE GUIDE TO CLIENT/SERVER PROGRAMMING USING OS/2 2.0

This second edition contains over 600 new pages on advanced client/server technology and 32 bit OS/2 features. Through easy to read, in-depth tutorials and 300 pages of working code the book shows you how to integrate databases, LAN's and Workplace Shell clients to build client/server applications on an OS/2 platform.

The book covers OS/2 2.0's base services-including threads, semaphores and NamedPipes; LAN protocols - including CPI-C/APPC, NetBIOS, Sockets over TCP/IP, NetWare Requester, and LAN Server; and database servers - including Database Manager and DDCS/2. Over 200 pages are devoted to Object-Oriented User Interface (OOUI's) using SOM and the Workplace Shell Class Libraries.

The first 200 pages, "a book within a book", is a very readable, general introduction to the OS/2 2.0 Client/Server platform. The last 900 pages are great for programmers at every skill level.

Look for this book and other OS/2 2.0 Library Books from Van Nostrand Reinhold at Barnes & Noble, Bookstop, Doubleday Superstores, and other fine technical bookstores nationwide, or to order call our Toll-Free number 1-800-926-2665 Dept. Z1375

VNR
*Publishing for Professionals Since 1848*

D1441

Join experts from around the world for the most important Software Development Conference of the year in New York City

IBM

# OS/2 Developer Conference & Exposition

## SHERATON NEW YORK HOTEL & TOWERS
## OCTOBER 18-21, 1992

This conference is exclusively designed for Developers. Now you can learn powerful new skills to enhance your OS/2 programming abilities. IBM has gathered some of the best speakers in the industry to teach you all you ever wanted to know about OS/2. Select from topics that enhance and upgrade your programming skills.

Join developers from around the world to share your ideas, skills and solutions for today's advanced OS/2 technology

**The conference features:**

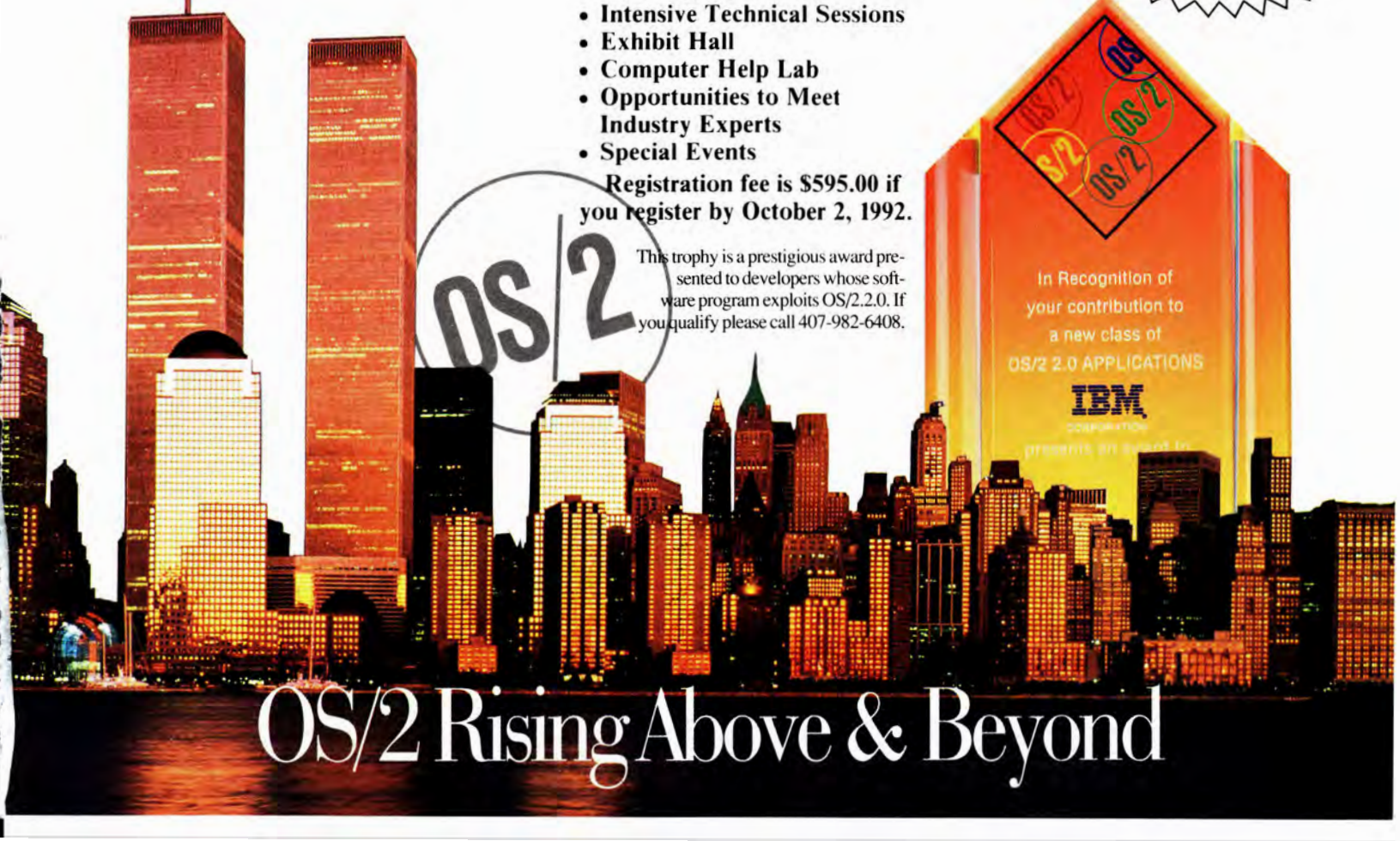
- **Guest Speakers**
  - **John A. Soyring, Director Software Development Programs, IBM Personal Systems**
  - **Lois Dimpfel, Director IBM PS Programming Center, Boca Raton**
- **Intensive Technical Sessions**
- **Exhibit Hall**
- **Computer Help Lab**
- **Opportunities to Meet Industry Experts**
- **Special Events**

**Registration fee is $595.00 if you register by October 2, 1992.**

For further information or to register, please contact the conference hot line 1-800-GET-OS20 (1-800-438-6720) USA & Canada or 1-203-261-6227 international.

**ATTEND AND RECEIVE OS/2 CD-ROM* & Borland ObjectVision**

This trophy is a prestigious award presented to developers whose software program exploits OS/2.2.0. If you qualify please call 407-982-6408.




# OS/2 Rising Above & Beyond

Performance

# OS/2 Productivity Through Multitasking

Pat Scherer

*by Pat Scherer*

*In the computer marketplace, OS/2 has been positioned primarily as a server platform; its prioritized multitasking scheme makes it an excellent choice for use on database, file, and communication servers. This article, however, focuses on how general OS/2 features can be used to improve productivity when installed on the workstations of individual users. It also compares OS/2's features to those of Microsoft Corp.'s DOS and Windows environments and presents a method for quantifying system performance benefits within a user environment.*

## THE VALUE OF MULTITASKING

While the OS/2 environment and Microsoft Windows over DOS™ allow users to view multiple applications, only OS/2 supports full multitasking. In addition to the obvious advantage of tasks continuing to execute in the background, OS/2 also provides a task prioritization scheme that allows excellent foreground screen response while running background tasks. Improved reliability for concurrent communication is accomplished with a higher priority for real-time or time-critical tasks, rather than the simpler DOS technique of time-slicing.

For long-time DOS users, making full use of multitasking requires breaking a number of habits. One such habit is watching tasks execute rather than working in another window (similar to watching the commercials taped along with the show on a VCR). A convenient feature of OS/2 2.0 automatically opens the same applications at startup that were active at the time of shutdown. This function encourages users to set up windows for regularly used applications rather than repeatedly loading and exiting programs. A "beep" or change in icon is a desirable application feature that alerts users to a change in status (such as transaction completion) when running tasks in full screen mode in the background. This feature is presently unnecessary in Microsoft Windows because tasks must be running in the foreground to complete. Another OS/2 feature not seen in Microsoft Windows allows users to watch the progress of applications in a background window while simultaneously working in a foreground window.

While the desire to speed up system response motivates users to upgrade to faster and faster systems, individuals rarely utilize more than 20% of the CPU of even a 10MHZ 80286 machine over a given hour's time. Multitasking speeds up task processing and more fully utilizes an investment in hardware.

One measure of multitasking is the overall productivity difference between a common set of tasks executed sequentially (as in DOS mode) and concurrently (in OS/2). This simple methodology can be tailored to any application environment. In one test, a set of common user scenarios were created from activities such as program compiles, file transfers, database, spreadsheet, and host interactions. Using a precise methodology to ensure repeatability, the time required to run the multitasking scenario was compared to the time required to perform the same tasks sequentially. The results of these benchmark tests, first executed on a 10 MHZ PS/2 Model 60 with OS/2 Extended Edition™ 1.0 and repeated with later releases of OS/2, showed the advantages of multitasking operating systems:

| TASKS RUNNING CONCURRENTLY | PRODUCTIVITY IMPROVEMENT | FOREGROUND DEGRADATION |
|---|---|---|
| Async File Transfer + Database Query | 1.6X | 5% |
| Database Sort + Database Query | 1.7X | 5% |
| SDLC File Transfer + Database Query | 1.8X | 3% |
| Database Sort + 3270 Screen Interaction | 2.0x | 0% |
| SDLC File Transfer + Async File Transfer + Database Sort + 3270 Screen Interaction | 2.8X | 5% |
| Async File Transfer + Database Sort + Compile + Spreadsheet Macro | 3.1X | 5% |
| SDLC File Transfer + 3270 File Transfer + Async File Transfer + Database Sort + Compile + Spreadsheet Macro | 2.9x | 5% |

*Table 1: Productivity gain due to performing tasks concurrently*

- Multitasking resulted in a productivity gain of just under two to over three times that of the sequential environment, depending on number and combination of tasks run.
- Background tasks did not have a significant impact on the performance of tasks running in the foreground.

Table 1 gives a more detailed look at the results of some of the user scenarios.

Actual productivity gain is a factor of the number of concurrent processes and the degree of resource contention on a system. The proportionately lower productivity gain in the last scenario indicates that resource (CPU) saturation occurred with the addition of the last two tasks. This brings up a flaw in the methodology as it applies to an actual user scenario: in implementing the test cases, all tasks were run from command files or macros without including "think time" to simulate user response on the foreground task. The multitasking performance of the more intensive scenarios was gated using the 10MHZ CPU; the addition of think time produces additional gaps in CPU usage that allow for faster processing of background tasks. Similarly, use of a faster processor would result in greater gains, as "think time" and file transfer rates remained constant.

Those familiar with performance testing know that benchmark results do not always translate into real productivity gains for the user. The most effective way to evaluate OS/2's usefulness is to apply it to a specific user environment and implement practices that take advantage of its multitasking features. Following are two examples of highly productive uses of OS/2 and tips for applying multitasking to enhance productivity.

In the first example, a series of 12 computer simulations were run in six concurrent overnight sessions. A test run of a single simulation took just under three hours to run, with an over 300% gain in turnaround time for the completed job.

In a second example, OS/2 was used to review, enter, and process stockmarket transactions while simultaneously downloading new market data with an asynchronous modem. Before installation of OS/2, the downloading process made the computer inaccessible for anywhere from one half to three hours, forcing users to do market analysis late at night in order to be ready for the next day.

*Use of a faster processor would result in greater gains as "think time" and file transfer rates remained constant.*

## GETTING PRODUCTIVE ON OS/2

The following productivity tips can help users get the most out of OS/2:

- Don't wait needlessly for a lengthy process to complete in the foreground; switch to a new window and work on something else.
- Open a new window rather than exiting an application you are likely to access again.
- Create command files or macros to process regularly performed tasks.
- Bundle multiple tasks together with a command file and run them in the background.
- Consider running multiple instances of a program (as in the simulation program example) rather than making multiple sequential runs.
- Use OS/2 mode software whenever there is a choice between OS/2 and DOS versions. This gives you the full benefit of multitasking and reduces context switches between DOS and OS/2.

*Some of the greatest productivity enhancements can be realized by customizing user interfaces to make multitasking more "natural"*

Numerous articles provide excellent advice for developers creating efficient OS/2 applications. Most of these articles focus on good coding practices and taking advantage of OS/2's internal features to minimize the length of frequently used code paths. As users become more accustomed to multitasking, some of the greatest productivity enhancements can be realized by customizing user interfaces to make multitasking more "natural." An interface can

- Alert the user when a background process completes
- Allow the user to batch a set of tasks
- Allow the user to select points within a batch process to view intermediate results and enter additional instructions.

OS/2 provides many of the building blocks for creating a system that is always available to process requests and yet fully utilizes a user's investment in processing power. Realizing the full potential of this system requires a creative approach that moves away from the realm of linear thinking—and from the constraints of DOS.

## EVALUATING PRODUCTIVITY GAINS OF MULTITASKING

The following method, "Task Concurrency Ratio for Performance Evaluation," from the *IBM Technical Disclosure Bulletin* (March 1990), provides a basis for making performance comparisons between sequential and multitasking environments. It includes instructions for setting up a test, an algorithm to normalize data, and a format to analyze and present test results.

### *Step 1: Identifying User Scenarios*

The background applications used for OS/2 Extended Edition testing were file transfers over 3270, SDLC, and an asynchronous communications modem, an assembler program compile, and a database sort. Foreground activities included a spreadsheet macro, an SQL database macro, and host screen scrolls via a 3270 or asynchronous connection. For accuracy and repeatability, foreground activities were implemented using either macros or, in the case of screen scrolls, a special keyboard driver. Foreground activities did not include user think time.

A less rigorous test can be done without a keyboard driver by identifying frequent activities and grouping the activities into clusters to be executed simultaneously. A script is written for each foreground activity and performed with each test. (Presumably, if the foreground activity is performed frequently, changes in think time due to repetition of the script should be minimized.) A command file is used to drive multiple iterations or an endless loop of each of the background activities. Data should be output to a virtual drive to eliminate disk activity resulting from the benchmark. A system timestamp is inserted between each iteration:

```
@ECHO Example Test Command File (TASK1.CMD)
:
:LOOP
@ECHO BEGIN EXECUTION
@PROMPT $T $G
RUNTEST
@PROMPT $T $G
```

```
@ECHO END EXECUTION
@IF NOT EXIST STOP GOTO LOOP
@GOTO EXIT
:EXIT
```

*Step 2: Measuring the Sequential Response of Each Task*

Multiple iterations of each background task are run with output redirected to a virtual drive, using the command

```
[virtual drive]:TASK1 >
[virtual drive]:TASK1.DAT
```

The background task is terminated with Control-Break, and the foreground script is executed with start and stop times recorded.

*Step 3: Measuring the Response of Each Task in a Concurrent User Scenario*

A separate window is created for each background task to be measured in a user scenario. Multiple iterations of each task are run and the output is redirected to separate files. (All background tasks should now be running concurrently.) The foreground script is then executed and start and stop times recorded. Upon completion of the foreground task, all background tasks are terminated. The start and stop times of the foreground task are used to identify a "window of concurrency" in which all tasks run simultaneously. All data outside of this window is discarded.

*Step 4: Evaluating the Data*

Any differences in the foreground execution of the concurrent and single task scenarios should be noted (for example, jerkiness or unresponsiveness from the mouse or cursor). You can then calculate the average elapsed time for a single iteration of each task in each scenario. Data can be evaluated with the Task Concurrency Ratio, shown in Table 2.

In this ratio, N equals the number of concurrent tasks, Tsi equals the average elapsed time for a single iteration of task i to run alone, and Tci equals the average elapsed time for a single iteration of task i to run concurrently.

This formula provides a partial sum for each task (Tsi/Tci) and a ratio for the scenario (R). Several performance characteristics of the scenario can be derived from this information. Normally, the partial sum of each task will be between zero and one. If the partial sum for a given task equals one, no degradation occurred in task performance as a result of multitasking. The lower the value of the partial sum, the greater the degree of performance degradation. Of particular interest is the partial sum of the foreground task, which should show little or no degradation over its execution in a single-task environment. The relative value of each of the partial sums is a good indication of the fairness of time slicing and task priorities within the concurrent environment. It is also a good indicator of resource contention.

$$R = \sum_{i=0}^{N} (Tsi/Tci)$$

*Table 2: Task concurrency ratio*

The ratio (R) generally has a value between zero and the number of concurrent tasks run (N). R indicates the productivity gained through multitasking. If R = N, no performance degradation occurred within any of the tasks. If R is greater than one but less than N, some performance degradation occurred as a result of the concurrent environment, but the overall throughput exceeded that obtained if all tasks had been run sequentially.

**Pat Scherer,** *IBM Personal Systems Line of Business, 11400 Burnet Rd., Austin, Texas 78758. Scherer joined IBM in 1980 as an industrial engineer. She holds an M.B.A. in industrial management and a B.A. in microbiology from the University of North Texas and completed the Graduate Certificate Program in computer science at the University of Texas, Austin in 1991.*

## REFERENCES

"Task Concurrency Ratio for Performance Evaluation," *IBM Technical Disclosure Bulletin,* Vol. 32 No. 10a (March 1990): 336-338. (IBM Doc. AT888-05-24)

Tools

# AM/Workplace: Building Client/Server Applications

Laurence Shafe

*AM is a high-productivity graphical programming environment that enables teams of programmers to work together to develop complex client/server applications.*

*by Laurence Shafe*

*AM/Workplace™, a significant new version of the AM development environment, enables teams of developers to quickly develop 32-bit applications that use all OS/2 2.0 graphical features , including direct manipulation, notebooks, and containers. AM/Workplace also has strong support for communications and database programming, making it ideal for developing client/server applications with an object-oriented user interface. Client/server, a well-publicized architecture for application development, can achieve returns on investment of over 300%. With client/server, users become more effective at their jobs due to applications that combine the personal flexibility and responsiveness of the PC with instant access to corporate and external information. Client/server applications are difficult to build, as they combine GUI programming, communications programming, and database programming, each a separate and demanding skill. AM is a high-productivity graphical programming environment that enables teams of programmers with various skills to work together to develop complex client/server applications.*

Client/server architecture works by separating the two basic components of a commercial application: user interaction and data manipulation. Users benefit from a highly responsive application, while corporate data is secure on a dedicated server that can be simultaneously accessed by multiple users. A server may be a workstation, minicomputer, or mainframe.

It has been estimated that 80% of a typical data processing department's time is spent on minor enhancements and maintenance of existing applications. Part of this problem arises from the need to port applications to more powerful hardware. An original objective of IBM's Systems Application Architecture™ (SAA) was to minimize the cost of porting applications between hardware platforms. The biggest problem was porting from a dumb terminal to a graphical user interface (GUI) PC, as it requires the application to be redesigned. By designing a client/server solution, the back-end database becomes easier to upgrade from one hardware platform to another. But to achieve this objective, existing applications must be rewritten. It becomes important to minimize the cost of this investment.

AM is a high-level programming environment that reduces the cost and speeds up client/server application development by teams of programmers. AM/Workplace, a new product from Intelligent Environments, creates 32-bit CUA 1991 applications.

AM, first shown at the IBM COMDEX Fall 1988 exhibit, became the first development tool launched for Presentation Manager. It has continued to stay at the forefront of OS/2 development since then, and now has over 1,000 customers developing major applications. Over one billion dollars in business is now committed to projects based on AM worldwide.

AM improves overall development productivity due to its unique graphical programming environment and dictionaries of reusable code. AM allows the logic of an application to be represented as lines of free text joined by logical connectors. This makes AM very easy to learn, as it is largely syntax-free. Its simplicity and clarity reduce the cost of enhancements and maintenance.

AM improves the quality of a finished application, allowing developers to work alongside users. Developers can respond to users' comments on screen layout and interaction and make changes within seconds; AM requires no compile phase. Its graphical language also enables users to comment directly on the correctness of program logic. Any procedure or data structure can be made public to other developers and can be pasted from a dictionary during development. This function facilitates code reuse and team-oriented development.

AM v. 3 was launched at the end of 1991 with support for MDI and CUA 1989. AM/Workplace, launched this year, is a 32-bit product with full support for the Workplace Shell, CUA 1989 to CUA 1991 migration, and support for all major SQL databases. Table 1 lists the current AM products.

AM/Workplace includes the following features:

- A graphical development environment with text, structure, outline, and dictionary views of the procedural logic and an integrated trace and debug system
- The ability to develop dynamically linked modules from libraries of code and create .EXE files for production applications.
- A wide range of database and communications support.
- Automatic migration of CUA 89 applications to CUA 91
- Full support for multitasking, modeless windows, direct manipulation, containers, notebooks, and other features of the Workplace Shell.

## BUILDING SERVER ACCESS

This article considers the implementation of both ends of the client/server solution, first looking at server and server connection options and then at graphical user interface considerations. There are several ways to link to the server, including:

- Screen renovation for existing mainframe applications
- Direct communications programming
- Remote program calling
- Distributed database access.

*Screen Renovation*

Screen renovation involves adding a front-end application to an existing 3270- or 5250-based application with a GUI. Strictly speaking this is not a client/server solution; it has, however, proved to be an effective way for organizations to add value to existing applications with minimum disruption to the mainframe.

The AM HostView/Workbench allows front-end additions to be developed with a point-and-click technique. The mainframe application is displayed in a window controlled by HostView, a developer clicks on fields required in the graphical application. Hostview automatically generates the AM code required to access or update the field.

This type of application suffers from inaccurate validity checking; if the application gets out of step with the mainframe, invalid information can be written over the corporate database. But with appropriate data integrity

**AM PRODUCTS**

| | |
|---|---|
| **AM**™ | for 16-bit development |
| **AM/Workplace**™ | for 32-bit development |
| **AM/Delivery**™ | for creating 16- and 32-bit .EXE files |
| **SQL/Workbench**™ | supports DRDA and static SQL development for OS/2 DBM, SQL/400, SQL/DS, and DB2 |
| **CP/Workbench**™ | supports transaction-based communications using APPC, CICS OS/2, and other protocols |
| **HostView/Workbench**™ | supports 3270 screen renovation applications |

*Table 1: AM products*

and screen status checks at both ends, these dangers can be minimized.

### *Direct Communications*

The standard solution for LAN and wide area network communications is advanced program to program communication (APPC) with the LU6.2 protocol.

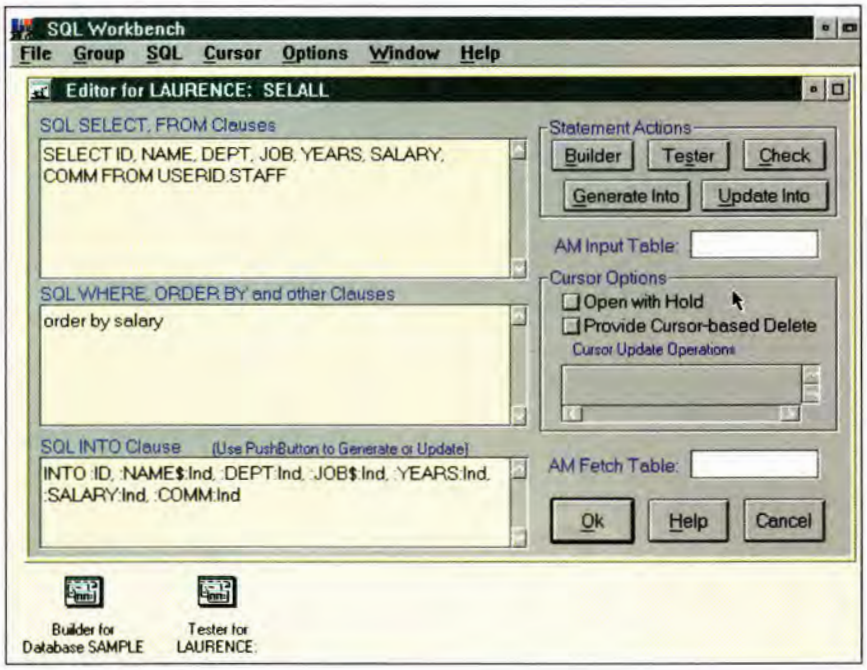



*Figure 1: SQL/Workbench*

The AM Cooperative Processing/Workbench provides three levels of APPC programming support within AM. At the lowest level, the APPC API can be called directly. The next level is a record-based set of verbs that provide automatic data-type conversion. At the highest level, there is a single function that performs type conversion, sends a complete AM transaction of mixed data type and returns a transaction in response.

The Workbench products store program components in a database, or repository. CP/Workbench, for example, stores transaction details and buffer definitions in the repository. Each transaction and buffer is named, and the program refers only to the name, not the details. The details can be amended independently of programs, which saves time if only a few parameters on a series of transactions need to be changed throughout a suite of programs. It is also possible to change the communications protocol without changing the program.

### *Remote Program Calls*

With APPC, a remote program can be started automatically; it can then communicate with the workstation. An alternative is to use CICS to control program activation and communication. The advantage to this method is that more organizations have experience with CICS programs than with APPC code.

CICS provides a complete transaction-based architecture for wide area network communications. This involves running CICS for OS/2 on the client machine to provide a program-to-program calling mechanism that is independent of the program location.

AM CP/Workbench supports many communications protocols, including APPC and CICS , all using the same high-level function set. With CICS, functions pass a data table in the CICS program communication area; the return data is then picked up and stored in another table.

### *Distributed Database Access*

AM supports IBM's Database Manager, SQL/Server™, Oracle™, Sybase, DB2™, SQL/DS™, and SQL/400™ for the AS/400™. DB2, SQL/DS, and SQL/400 are supported through Distributed Database Connection Services/2™ (DDCS/2). DDCS/2, a new IBM product for OS/2, allows an SQL request on a client machine to be routed to either Database Manager, DB2, SQL/DS, or SQL/400. These databases communicate using distributed relational database architecture (DRDA), a standard developed by IBM. DRDA's open APIwill soon be supported by other database manufacturers.

SQL/Workbench has been enhanced to support all SAA databases using static SQL. IBM relational databases support two types of SQL access, dynamic and static SQL. Static SQL is compiled SQL, suitable for production applications. Dynamic SQL is suitable for management information systems in which a user can create an SQL request on the fly.

AM has full support for both types of SQL, including multitasking data access, cursor control, and prefetching of information. The SQL/Workbench can store SQL statements in its repository, edit them, and compile them into a static SQL request. Each request is named so it can be invoked within the program. This approach ensures that SQL statements can be modified independently of programs, an ideal mechanism for tuning database access after the application logic has been completed. Figure 1 shows a sample screen from the SQL/Workbench. The Workbench provides many other features, including data type conversion, prompted SQL generation, and the testing of SQL statements with automatic report generation of the data selected.

DDCS/2 changes the way data is accessed from host databases. Because security and performance are critical when reading and writing host databases, static SQL is essential for this process. Static SQL provides the performance and security required for a production application; because it is predefined, it is possible to predict loading and performance of a finished application.

Specialists responsible for database access are known as database administrators (DBAs). SQL/Workbench enables a DBA to control how a database is accessed without getting involved in the programming.

## THE AM/WORKPLACE PROGRAMMING ENVIRONMENT

The server end of AM programming is largely handled by the Workbench products. Window handling, calculation, and logic are written in the AM/Workplace programming environment.

Over the last 40 years, literally thousands of programming languages have been created. The creators have usually focused on the features of the language rather than its usability, resulting in many sophisticated languages but few that are easy to learn. Most languages contain a wealth of features that take months of intensive training to use. As a result, many organizations stay with languages they know, such as COBOL, first defined in 1954. The advent of the graphical user interface offers an opportunity to create a new type of programming environment, referred to as visual programming. By fully using the graphical environment, AM provides the power of a conventional language yet is easy to learn. AM creates program structures that are easy to modify and maintain.

The AM programming language is logic based, reduces programming errors, and creates easy-to-maintain programs. Its graphical interface is used to display multiple views of a program, including structural views, an outline, procedural logic, dictionaries, and menus. Each line of the program is tagged with icons showing procedure type.

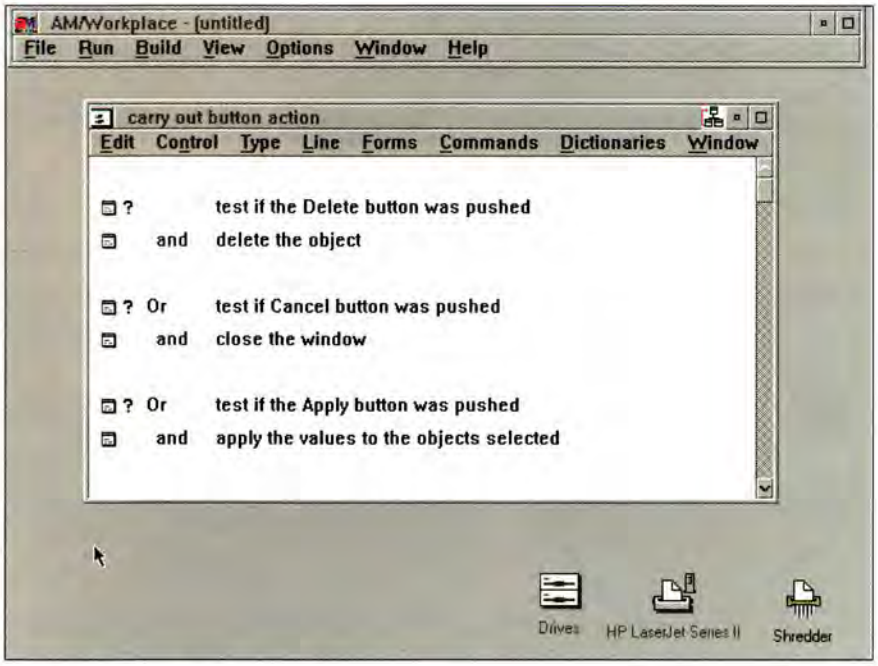



*Figure 2: An example AM procedure*

The AM language is based on a process of functional decomposition. Starting at the top, each functional unit, or procedure, is defined by entering its logic as lines of descriptive text. Tied to the interactive graphical user interface of AM (which provides dictionaries, graphical displays, an outliner, form filling, and so on), it offers rapid, simplified entry of complex logic.

Any line of descriptive text can be expanded into a procedure that consists of lines of free text. At the lowest level, a command can be

inserted from a pull-down menu of possible commands. The list of commands is kept precise to simplify learning.

The flow of control through an AM program is determined by procedures' success and failure. This mechanism can be used to implement a switch statement, as shown in Figure 2. As each procedure line is executed, it will either succeed and step on to the next line or fail and jump immediately to the next alternative. The flow of control through the program is determined by these successive tests.

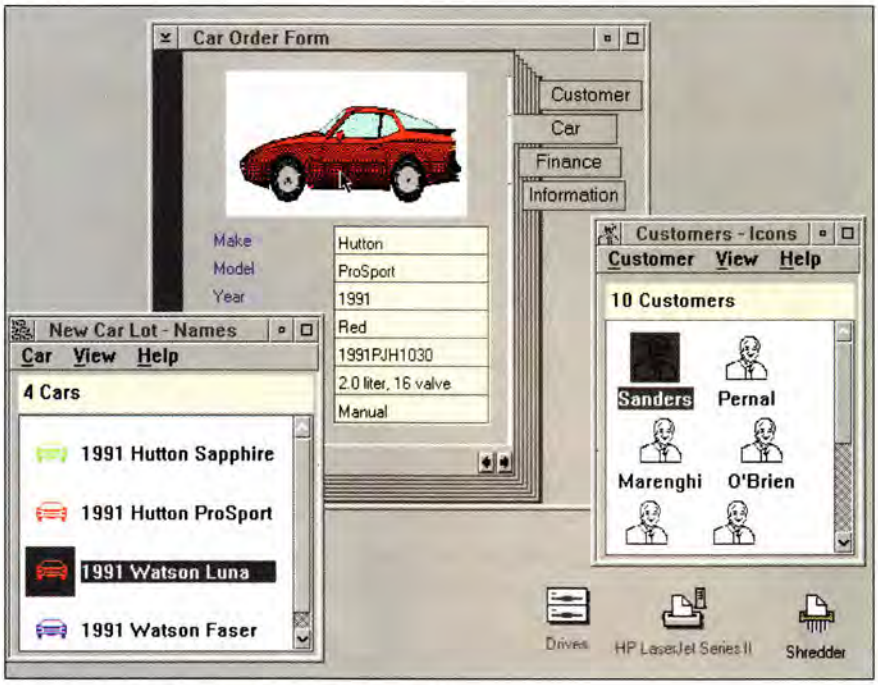



*Figure 3: CUA 1991 car dealer system built with AM*

To assist with programming, AM analyzes the logic structure of each line as it is entered. The line's logical outcome is then displayed as an icon alongside the line. The programmer is guided by the logic analyzer as he or she writes the program.

Programs are created by defining procedures and painting windows using the integrated window painter. Programs are written as a set of modules that are dynamically linked at either load time or when a procedure is called. Any procedure or data structure can be made publicly available, then used by other developers who access procedure and data dictionaries.

Applications are built up from multiple modules that can be viewed either as a tree structure (the process structure or functional decomposition) or as an indented outline (the program view). Modules are sub-divided into procedures; each procedure is a single logical unit that is described by the programmer and can be viewed and edited visually.

An AM module is able to make any of its procedures publicly available and hide (or encapsulate) the others. One model can use procedures in another; they can communicate by message passing or by means of publicly available data structures.

An AM application is self-documenting, in the sense that AM encourages a free text description of procedures. Because each procedure is self-describing and has a simple logical structure, it is easy for another programmer to understand what a procedure is doing.

The benefit of this approach is that AM programs have few undocumented anomalies, or bugs, and are easily maintained and extended. They also offer author independence: AM programs are self-documenting and their visual nature makes them easy to understand and modify.

AM not only develops Workplace applications, it is a Workplace application. AM's development environment provides multiple modeless secondary windows and a variety of views, program objects, and actions for the programmer. The logic entry and editing of an AM program is executed within multiple edit windows, with different levels of the logic accessed with the mouse. Some program operations, such as assigning values to variables and testing results, are prompted with pop-up, context-sensitive forms. User entry forms are created and edited with a form painter in the same environment.

Within the window editor, PM controls such as containers, list boxes, radio buttons, entry fields, and so on are defined with a full-function PM interface. With definition capabilities and using the mouse to click-and-drag, a full set of alignment and duplication features make rapid screen development easy. Program variables are defined or selected from

the AM variable dictionary and are automatically available to other AM program statements.

UNUM Insurance, the world's leading provider of long-term disability insurance, measured AM productivity with function point analysis to determine how many functions were programmed in a programmer-month. The average mainframe project had previously been measured at 6.5, while the best recorded number was 18. The AM project recorded over 50, including the time needed to get programmers up to speed on the workstation, training on AM, and the normal problems associated with first-time use. The organization is now planning to provide all programmers with AM over the next two years.

AM applications can be run directly from the development environment with no compilation or binding delay. An application can be run in trace mode so every procedure is animated and can be executed either a line at a time or as a complete procedure.

A large part of the productivity of AM comes from its ability to reuse code by storing procedures in libraries. When a developer loads a library, all its public procedures are copied into the local dictionary. The developer can then paste the required procedures into the logic structure.

When a program finishes its test run, the development parent window is displayed again, allowing the developer to make immediate changes to the program. While tracing the program, a developer can leave execution and edit the statement or window being executed with a pull-down command.

Rapid prototyping and short test cycles are simple with AM and are essential for dealing with the wide variety of look-and-feel options available with a graphical user interface.

## BUILDING THE WORKPLACE USER INTERFACE IN AM

OS/2 2.0 offers a new user interface that is more intuitive, easier to learn, and more effective than the old Windows-style interface. The major thrust was to simplify the visual desktop metaphor, reducing the number of subsystems and performing actions by dropping object icons onto one another.

Commercial applications are used by a wide range of people, from individual users who use PCs as part of their daily activities to single-task users. PC users tend to use a variety of utilities and productivity tools and quickly become conversant with the Workplace features. Single task users, in contrast, use their PCs more narrowly. For example, a banking system that gives advice on loans and accounts is used by a bank teller or customer. In this case, the computer screen must be self-explanatory, is likely to be form-based, and can be more modal in style. These user requirements involve a number of basic components:

- Objects and their views
- Windows (modal and modeless)
- Containers
- Direct manipulation by dragging and dropping icons
- Notebooks.

These new features can be used to build a practical application using AM; the following example is from the IBM CUA 1991 manual, and is available on disk from Intelligent Environments.

The application allows a salesperson to create an order form using information on a customer and car. Customers, cars, and worksheets are represented as objects. There are two folders, one containing customer information and the other listing the cars in the dealer's lot.

To build the application, we must implement objects and their relationships. The following description concentrates on user interface and interaction, rather than on database issues, which are touched on at the end of this section.

Let's start with the car lot. This is a folder containing all cars in stock. This folder, normally on the workplace all day, is programmed in AM by creating a module that displays the car lot window. A new control, called a container, has been added to PM and

*AM applications can be run directly from the development environment with no compilation or binding delay.*

is provided in the AM window painter. In the AM program, the container is associated with a data table containing all the cars to be displayed. The container is added to a window by painting the container the required size with the integrated window painter. Each control drawn on the window pops up a form on which to enter information describing the control; for example, its color, the font to use, the values to display, and procedures to call when a message is received. The container can also be marked to resize with the window.

When the application is run, AM handles all events resulting from the user's interaction with the window. Each event is associated with a message; for example, "mouse up," "mouse down," "double click," and "object dropped." AM filters these messages and then checks to see if an AM procedure has been defined to handle the general event. For example, when defining a check box, the developer can add a procedure called when a user clicks on the check box. Minor messages, such as those describing mouse movement or window repainting, are handled without writing any procedures.

*AM applications can be written in any style from completely procedural to full event or message-driven.*

In a low-level language such as C, the programmer must write code to handle all messages. AM handles everything by default; the programmer only needs to add procedures when he or she wants to perform specific application logic.

AM applications can be written in any style from completely procedural (a sequence of modal windows with AM handling all the messages automatically) to full event or message-driven (with the program responding to messages). The CUA '91 object-oriented user interface encourages all applications to be message driven and modeless. To the user, this means applications that put him or her in control and are responsive to clicks and direct manipulation.

Let's return to the car lot window. The main window area is defined with the window painter. At the top of the window, below the menu, is the status area. This contains the time of the last display, the number of cars matching the `include` criteria, and the total number of cars in stock. The car lot menu is defined with the AM menu builder; each choice is associated with a procedure. The help menu can either use the help features built into AM or link directly into the OS/2 Help Manager.

In the menu is the `include` option, which displays a secondary window that lets the user enter selection criteria. Cars matching these criteria are then read from the database and displayed in the car lot window. The database always represents the up-to-date corporate data view, while the information on the desktop is the user's view. Whenever a user performs an action on a local object, the application checks with the database for up-to-date information.

The central part of the car lot window is a container, associated with an AM data table that responds to messages by calling an AM procedure. A table is an AM data structure consisting of named columns of mixed data type and numbered rows. It corresponds to an SQL database table and can be filled by a single SQL `SELECT` command.

The container displays data in the table as one of a number of standard views. The icon view consists of rows of icons with a name underneath each icon. The name is taken from a specified column of the data table, and each row of the table corresponds to a single icon. A container needs to respond to a number of messages with explicit application logic. Two described here are double clicking and direct manipulation.

A user may double click on an object in a container that then calls an AM procedure. For example, a user double clicks on one of the cars in the car lot object. The AM procedure is passed the row number of the object. It can then start another window and pass that window the details of the car.

The created window is an AM module that displays details of a selected car, such as a picture of the car and a number of output fields. The information displayed may also require an SQL database to pick up information not displayed in the container, or may use a graphic or video clip. A similar method is used to create the customer list .

### *Direct Manipulation*

Finally, a user can interact with objects by direct manipulation. He or she picks up one

object using the right button on the mouse, drags it to another place, and drops it by releasing the button. Often, the user will drop one icon on top of another.

AM provides automatic handling of direct manipulation but allows developers to add modifying procedures. For example, objects can be moved, copied, or deleted from one AM container to another, or to the Workplace Shell, without any programming in AM.

Overriding the default functionality requires a definition of procedures. There are two sides to drag and drop, the object being dropped and the object dropped upon. When a drop occurs, AM calls a procedure in each object. The procedures receive messages from other windows, icons, and threads, communicate between the objects, and then carry out any necessary action.

For example, dropping a customer onto a worksheet adds that customer's details to the worksheet. Customers are objects displayed in the customer-list container. When one is dropped onto the worksheet, an AM procedure is called in the customer list and worksheet object. The worksheet object picks up a reference to the particular customer (such as account number), reads customer information from the database using SQL, and then displays the information in the worksheet.

If a user drops an AM object onto a device such as the printer, the source container is notified and performs the necessary action, such as printing a document. If necessary, the application displays a window requesting additional information such as the number of copies to be printed.

If an AM object is dropped on a non-AM application, then the AM program passes details of the object to that application using dynamic data exchange or a standard ASCII format. For example, if an AM object is dropped into a Workplace folder, a file containing a text description of the attributes of the object is automatically created. This file can later be dropped onto the AM container, which will recognize the format and automatically add a new line to the AM table. This is an efficient way to move from an SQL database to an AM table to a container to a Workplace folder and back again. Validation and security checking can be added between any of these steps.

To restrict which objects are allowed to drop where, AM defines classes of objects on which a particular object can be dropped. If the object is moved over an object of an unallowed class, the "no entry" icon is displayed.

## USER OBJECTS AND CORPORATE DATA

The application developer must both create a good user interface and ensure the security of the corporate database. OS/2 2.0 Extended Services with Distributed Database Connection Services (DDCS/2) lets an AM application update IBM's Database Manager, DB2, SQL/DS, or OS/400 databases. AM can also access and update Microsoft's SQL/Server and Oracle databases.

*The application developer must both create a good user interface and ensure the security of the corporate database.*

The AM applications are the interface between objects in the workplace and rows in the database and must model the user's view of the desktop and the corporate database.

The first step in creating a database is to perform a detailed data analysis of a business's requirements, identifying basic data entities and suggesting a data structure of tables and views. Some views will become containers with the objects representing rows of the database. Each row corresponds to an object in the container shown as either an icon or a line of information, depending on the view. Objects can be examined in more detail and edited by displaying a notebook with pages containing fields for the object or row selected.

How the objects act and interact with each other is determined by the business's requirements and the needs of the user. In one possible model, the desktop is seen as the top of the user's desk and the database is a central filing system, with information on the desktop being a copy of the information in the filing system. For example, if a user displays a window containing details of a particular car in the car lot, it is a copy of information on file. Many sales people can therefore display the same details at the same time. However, if one sales person sells a car, it is no longer available for others to sell. This can be accomplished if

the application checks with the database each time a car is sold. As with an airline seat reservation system, an available seat (or car) may be taken by the time a request is entered.

If an object is deleted from the database, the object representing it on the workplace is also deleted. Alternately, a local copy can be kept for reference purposes although the original has been deleted from the database.

There are many ways to model the relationship between objects on the desktop and in the database. The method used depends, among other things, on the rate at which the database changes. For example, a customer list changes slowly over time and the car lot changes only as cars are sold and new cars arrive. In these cases, the database needs to be checked only at critical stages in the application. However, because of the time a sale can take and the importance of each sale, salespeople could be provided with the number of other customers interested in the same car. At some stage during the negotiation, a car could be reserved to prevent a sale to another customer.

*A key to success with AM is its SQL database access capabilities.*

## ACCESSING AND UPDATING SQL DATABASES

A key to success with AM is its SQL database access capabilities. SQL commands are executed within AM to retrieve selected rows, insert new rows, update existing rows, or delete rows.

There are three levels of SQL access in AM. At the first level, there is an integrated `SQL Request` command that supports SQL access to Database Manager, SQL/400, SQL/DS, DB2, SQL/Server, Sybase, and Oracle. This command is simple to use yet provides full insert, select, update, and delete access. The next level is a set of functions that provides cursor handling, independent control of up to 10 SQL tasks, prefetching for large selects, database create, logon, and message control.

Finally, a separate product called SQL/Workbench supports static SQL for Database Manager, SQL/400, SQL/DS and DB2 using DDCS/2. SQL statements, held outside the program code in a Database Manager repository, can be combined into groups, each associated with a programmer and a project. As statements are edited, a version control system keeps track of the reason for changes and the time and date they were made. The Workbench compiles groups directly using the precompiler services of the appropriate database. The Workbench also includes an SQL builder that prompts with valid SQL parameters and a test facility that runs the SQL statement against the database and produces an on-screen report that verifies the SQL is correct.

For developers, accesing and updating a host database is now as simple as using a local database. With AM, a two-line program that reads and displays data from DB2 can be entered and run in under 60 seconds. Developers can include SQL access by selecting the `SQL Request` pull-down menu from the main development environment. This prompts the developer to enter an SQL statement.

For `SELECT` statements, the full select set (result rows) is returned to an AM table, a special data structure similar to the relational table model. Each AM table has a name that describes the entire table and any number of columns consisting of AM text or numeric variables. These columns are then organized internally into rows and referenced by the programmer by row number and column name. All data analysis and type conversion is performed automatically by AM.

For convenience, AM maps AM table column names to database column names so the AM table can be filled from the database without explicit programming. This can be overridden with the `SQL INTO` facility. The results of an SQL select can be viewed with an AM list box or container. This procedure requires no explicit programming; the programmer simply types the name of a table while defining the control.

## CREATING SPECIALIST FUNCTIONS IN C

Any special interfaces, such as those that support a nonstandard communications protocol, can be written in either 16- or 32-bit C with the developer's toolkit supplied with AM. This toolkit provides a table-driven method to define new AM functions that can

be included transparently in the standard AM function dictionary. As AM is a 32-bit product, 16-bit C interfaces are supported through a "thunk" layer that automatically translates function calls between 16- and 32-bit formats.

## TESTING AND COMPLETING THE APPLICATION

At any time during development, an AM application can be tested by selecting the **Run** pull-down menu from the main development menu. The application under development is executed as if a user were executing it. There is no need to complete all defined procedures; at run time these appear as a message box to the developer, who can continue the application by responding affirmatively. Developing graphical user interfaces benefits significantly from the interactive nature of AM. Developers can design the PM screen and immediately test its look and feel during execution. Most AM application testing is not testing in the traditional sense of finding bugs, but rather it is an iterative enhancement of the user interface and application logic.

AM has an integrated trace and debug system that includes single stepping through a program, executing to a breakpoint or until an expression succeeds, and full access to program code and dictionaries. When an application is completed and ready for delivery, the AM Delivery System creates an .EXE program and one or more library modules. Once compiled, application modules can be distributed to users who execute them with the AM dynamic link library (DLL) support.

## SUMMARY

AM/Workplace enables teams of developers to quickly create client/server applications with an object-oriented user interface. Developed code can be stored in libraries and used by other developers on the team or within other applications.

AM/Workplace supports a wide range of communications methods and relational databases. Communications include support for EHLLAPI™, APPC™, CICS™ for OS/2, and Named Pipes™. It supports databases such as IBM's Database Manager, SQL Server, Oracle, Sybase, DB2, SQL/DS, and SQL/400. AM supports dynamic and static SQL with IBM relational databases. The product also supports a full range of OS/2 2.0 Workplace functionality including direct manipulation, notebooks, containers, multitasking, and object manipulation.

**Laurence Shafe**, *Intelligent Environments, 2 Highwood Dr., Tewksbury, Mass. 01876, (508) 640-1080 and Intelligent Environments Limited, Crystal House, P.O. Box 51, Sunbury-on-Thames, Middlesex, U.K., TW16 7UL +44 (0) 932 772266. Shafe is the CEO of Intelligent Environments; he founded the company in 1985 and has seen it grow into a multimillion dollar organization with offices across the U.S. and Europe. He wrote* OS/2 in the Corporate Environment, *from which parts of this article have been taken. Shafe has 24 years' experience in the computer industry; he holds a master's degree in computer science and a doctorate in artificial intelligence from the University of London.*

*Developers can design the PM screen and immediately test its look and feel during execution.*

### REFERENCES

For more information on AM/Workplace, call Intelligent Environments at (800) 669-2797 or (508) 640-1080 in the U.S. and +44 (0) 932 772266 in Europe.

## Multimedia

# Plugging Into MMPM/2

Dana Liburdi

Evi Larsen

*by Dana Liburdi and Evi Larsen*

*IBM Multimedia Presentation Manager/2 (MMPM/2) 1.0 is an extension that adds multimedia capabilities to the OS/2 2.0 environment. From an application perspective, MMPM/2 provides a device-independent programming interface and a run-time environment for multimedia applications. It also provides a standard interface for manipulating data and controlling devices.*

*From a device-driver perspective, MMPM/2's extendable architecture makes it possible to add new functions, devices, and multimedia data types and formats as technology advances, extending the capabilities of the operating system. Digital and analog video interfaces, for example, will be provided in future releases of MMPM/2. See Brad Noe and Bill Lawton's "Introduction to IBM Multimedia Presentation Manager/2" (IBM PS/2 Developer, Spring 1992, pages 72-77) for an introductory article on MMPM/2 interfaces.*

*This article focuses on the MMPM/2 Developer's Toolkit, which assists in the development of applications subsystems for MMPM/2. The Toolkit includes C and ASM bindings, libraries, sample code, and on-line reference material to help integrate multimedia into new or existing OS/2 applications and to develop new subsystems that extend the capabilities of MMPM/2. Use the Toolkit to create multimedia applications that include voice, music, CD digital audio, videodisk, video and still images, and graphics.*

## SAMPLE PROGRAMS

The MMPM/2 sample programs illustrate multimedia programming concepts and establish a basis for developers to create their own multimedia applications. The programs provide practical examples that span a range of multimedia concepts, from media control interfaces to interdevice-driver communication (IDC) interfaces. Sample source code shows you how to use multimedia controls and functions to create multimedia applications and subsystems. Each sample program serves as a template that can be modified to meet your application requirements.

The following discussion focuses on the implementation of MMPM/2 services and functions illustrated in the sample programs. The sample programs discussed in this article illustrate how to plug in to MMPM/2 from an application perspective.

## DUET PLAYERS I AND II

The Duet Player sample programs show how the media control interface layer of MMPM/2 controls multimedia devices for an application in a device-independent manner. These simple programs enable a user to play, pause, resume, stop, and change the volume of a song. The Duet Players also have multimedia integrated in their help information; they play an audio help file in response to help requests.

The Duet Player samples illustrate the programming concept of creating and controlling multiple multimedia devices, hence the term "duet." Each song played by the Duet Player is made up of two waveform files, which play together to make a duet—a song made up of two parts. In this sample, one of the waveform files consists of music and the other is a voice overlay.

The Duet Player opens a separate instance of a logical wave audio device to play each part of the duet. It then groups these two logical

devices into one logical unit. Instead of dealing with each part of the duet separately, the program obtains an ID for the unit (or group) and uses that ID to manipulate it. In other words, the application issues play, pause, and stop commands on the group as a single unit, instead of having to control each device in the group separately.

Duet Players I and II are identical except for minor differences in how the hardware devices are controlled. Duet Player I, shown in Figure 1, plays both parts of the duet on a streaming device, the M-Audio card. Duet Player II groups a streaming device and nonstreaming device (a device that handles data internally and does not stream data through the SPI subsystem of MMPM/2). The songs in both samples are divided into two parts; however, Duet Player II plays one part of the song on an M-Audio card, and the other part from a CD in a CD-ROM device connected to the system. This difference in hardware is almost totally isolated from the application by the media driver provided by MMPM/2.

The hardware differences between the Duet Player samples affect the applications very little; the MMPM/2 media drivers and stream handlers manage the hardware-dependent function. MMPM/2 significantly buffers applications from hardware dependencies.

The media control interface subsystem of MMPM/2, divided into two layers, provides the primary mechanism for application control of media devices. The top layer is a media device manager (MDM), which provides system management of media devices. The bottom layer consists of logical media devices or processors of media information. Applications interact with the media control interface (and thus with media devices) in two ways, through either a procedural or string interface. The procedural interface sends messages from an application to the media control interface using the `mciSendCommand` function. The string interface sends command-language statements from an application to the media control interface using the `mciSendString` function Both interactions are shown in Figure 2.

Figure 3 shows the source code used by the Duet Player to open a logical instance of the wave audio device for each of the duet parts.

This example illustrates the `mciSendCommand` API (i.e., the procedural interface).

The first parameter is a device ID, returned from the call in the following data structure. The second parameter is a flag to indicate the command to be executed. In Figure 3, we're sending an open command.

The third parameter specifies flags for MMPM/2 to use when processing the `MCI_OPEN` command. The `MCI_WAIT` flag tells MMPM/2 not

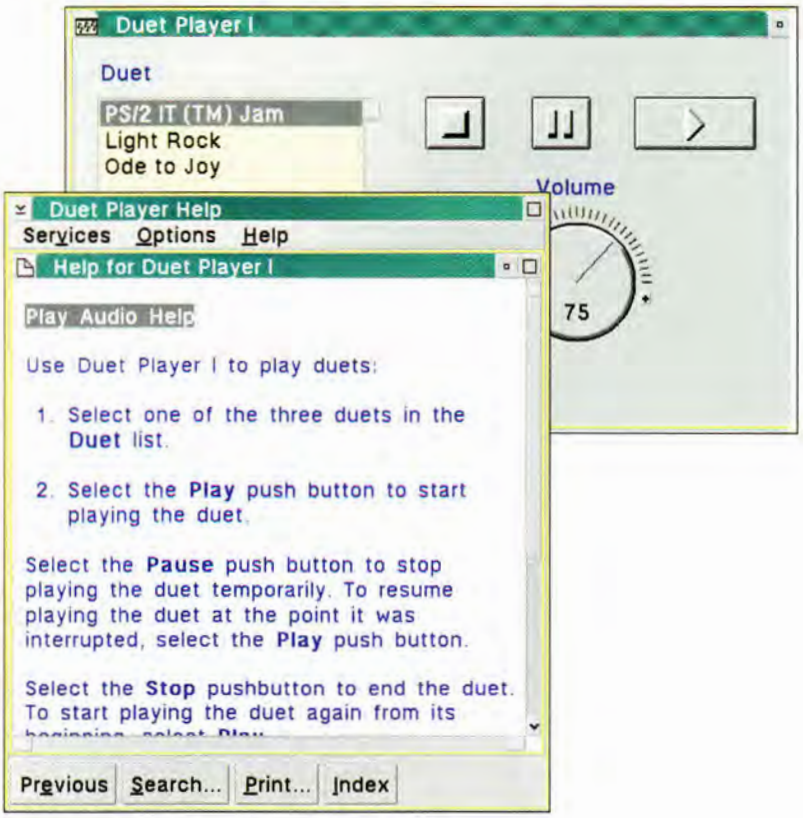



*Figure 1: Duet Player 1 window with IPF help panel*

to return to the application until it is done processing an open command. The `MCI_OPEN_ELEMENT` flag tells MMPM/2 to use the element name in the open data structure to choose which device to open. `MCI_OPEN_SHAREABLE` tells MMPM/2 to let other applications have access to the device while it is in use. `MCI_READONLY` tells MMPM/2 to open the object on the device so other applications can access the object while it is being used, as it will only be reading, not writing, the file.

The fourth parameter is a data structure dependent on the type of command issued. In

this case, issuing an open command means that we must use the corresponding open data structure. The device ID is returned here in the `wDeviceID` field. You can open the device in two ways, either by specifying the device in the device-type field of the structure (there are several defined device types in one of the public header files) or by specifying the name

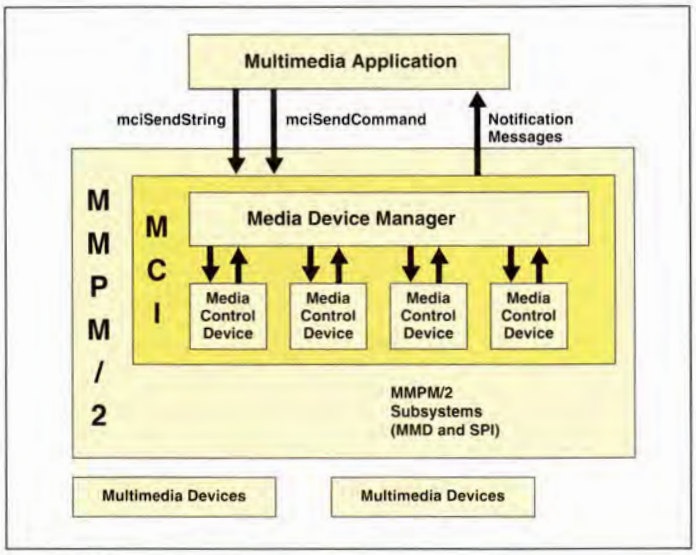



*Figure 2: Media control interface subsystem*

```
mopDuetPart.dwCallback       = (DWORD) hwnd; /* For MM_MCIPASSDEVICE*/
mopDuetPart.wDeviceID        = (WORD)  NULL; /* this is returned    */
mopDuetPart.lpstrDeviceType  = (LPSTR) NULL; /* using default conn. */
mopDuetPart.lpstrElementName = (LPSTR) aDuet[sDuet].achPart1;

dwError = mciSendCommand( (WORD) 0,
                          MCI_OPEN,
                          MCI_WAIT | MCI_OPEN_ELEMENT |
                          MCI_OPEN_SHAREABLE | MCI_READONLY,
                          (DWORD) &mopDuetPart,
                          UP_OPEN);
```

*Figure 3: Opening the waveform files*

of the element you wish to load (in this case, an audio file) on the device when you open it. If you specify the element name, as we do here, you can let the MDM figure out which device to open instead of passing the device ID. MDM does this using a default connections table specified in the INI file.

The first field of the `MCI_OPEN_PARMS` data structure specifies the window handle to which the application wants MMPM/2 to send `MM_MCIPASSDEVICE` messages. This message allows the application to participate in MMPM/2's device sharing scheme.

The last parameter is a value to be returned in a message parameter when MMPM/2 sends the application an `MM_MCINOTIFY` message.

Each part of the duet is played through MMPM/2 on a separate logical wave audio device, adding a twist to playing an audio file. If we had simply played the audio file opened in Figure 3, it would be processed through MMPM/2 using one instance of the logical wave audio media device. Instead, we aren't just playing one file through MMPM/2 using one instance of the logical wave audio device; we are playing two audio files through MMPM/2 together, each using its own logical wave audio device.

MMPM/2 makes it easy to control two logical wave audio devices simultaneously, or to control any group of multiple devices. An application groups the two devices together into one logical unit, and the MDM manages the devices, allowing the application to refer to the group as a single unit.

Figure 4 illustrates the `mciMakeGroup` function, which groups multiple devices into a single unit. An array is filled with the IDs of open devices to be grouped. (This is the device ID returned in the `OPEN` data structure from the `mciSendCommand` with the `MCI_OPEN` command specified.) Next, the `mciMakeGroup` function is called to build a group of media devices and return a handle to it. From now on, when the Duet Player application wants to play, pause, stop, or resume the group, it can refer to the group as a single unit using the group handle instead of having to manage each device separately.

The `MCI_NOPIECEMEAL` parameter in the `mciMakeGroup` API is a flag that tells MDM how to treat the device group. If the `MCI_NOPIECEMEAL` flag is not specified, the application retains control over the remaining devices in a device group if one or more devices in the group is lost. When this flag is specified, if any device in the group is lost, the entire group is lost, and each device in the group saves its state. In other words, the `MCI_NOPIECEMEAL` flag tells MDM never to split up the devices in a group. If one device must stop playing, all devices

stop playing. Similarly, if one device is able to resume playing, it can only do so if the other device can resume as well.

The last parameter in the API is a flag that indicates whether the group should be synchronized. In future releases, MMPM/2 will provide system management for the synchronization of devices in the group. For example, you might want to synchronize an audio device and a video device. The system management will free the application from any synchronization processing because it will be managed by MMPM/2. This is an exciting alternative to interleaving data to synchronize audio and video data streams.

Now that the group is created, it can be played using `mciSendCommand`, as shown in Figure 5.

The first parameter of the `MCI_SEND` command in Figure 5 is the group handle returned from the `mciMakeGroup` API. If you are playing a single device, specify the device ID (returned from the `mciSendCommand` with the `MCI_OPEN` flag) in this field.

The next parameter is the command to be processed by MDM, in this case `MCI_PLAY`.

The third parameter specifies that the application wants control returned to it immediately after issuing the command, instead of waiting until the command is finished being processed. On the previous `mciSendCommand` for `MCI_OPEN` , we specified the `MCI_WAIT` flag, indicating that we didn't want to come back to the application until the device was opened. But here, we want to come back to the application immediately, instead of waiting for the device to finish playing the audio files.

The `MCI_NOTIFY` flag tells MDM to notify the application when the group has finished playing. MDM sends a message to the window handle specified in the `dwCallback` field of the `PLAY` data structure, the next parameter in the API. In this case, the handle specified is the main window handle of the application, as shown in Figure 6.

Another multimedia concept used and illustrated by the Duet Player is device sharing. The Duet Player is well behaved; it lets other applications use the devices it is using and resumes use of those devices when a user focuses back on its window.

To ensure a well-behaved application, we must implement the correct device-sharing logic in our application. To start off the discussion about device sharing, refer to Figure 3. We have specified a window handle in the `OPEN` data structure callback field of the API (fifth parameter, first field). This tells the MDM where to send the `MM_MCI_PASSDEVICE`

```
awDeviceList[0] = wDuetPart1ID;
awDeviceList[1] = wDuetPart2ID;

dwError = mciMakeGroup( &wGroupHandle,      /* result handle      */
                        (WORD) NUM_PARTS,   /* count of devices   */
                         awDeviceList,      /* array of devices   */
                         MCI_NOPIECEMEAL,   /* flag               */
                         (DWORD) 0);        /* n/a - not synched  */
```

*Figure 4: Creating a group*

```
mppGroup.dwCallback = (DWORD) hwnd;
/* notify our window    */

dwError = mciSendCommand( wGroupHandle,
                          MCI_PLAY,
                          MCI_NOTIFY,
                          (DWORD)&mppGroup,
                           UP_PLAY);
```

*Figure 5: Playing the group*

*The Duet Player is well behaved; it lets other applications use the devices it is using and resumes use of those devices when a user focuses back on its window.*

message. In the case of the Duet Player, we specify the main window procedure handle. The `MM_MCI_PASSDEVICE` message will be covered later in this article.

In Figure 3, the `MCI_OPEN_SHAREABLE` flag is specified at the third parameter of the API. This indicates that the opened device can be shared with other applications. It is also possible to open a device for exclusive use by specifying a different flag in the field of the API. In this case, no other application could open or access the device while we had it opened. But because we are illustrating device sharing, we have used the `MCI_OPEN_SHAREABLE` flag.

Once you open a device as a shareable device, you can participate in device sharing. As we mentioned earlier, the `dwCallback` parameter of

the OPEN data structure tells MDM where to send MCI_PASSDEVICE messages. MDM sends an application one of these messages each time it gains or loses one of the devices it has open. The message is sent to the main window procedure of the Duet Player; when the Duet Player receives this message, it sets a global variable to indicate whether it has lost the use of the Duet's devices.

```
case MM_MCINOTIFY:
   wNotifyCode = (WORD) SHORT1FROMMP( mp1);
   wUserParm   = (WORD) SHORT2FROMMP( mp1);

   wCommandMessage = (WORD) SHORT2FROMMP( mp2); /* high-word       */

   switch (wCommandMessage)
   {
      case MCI_PLAY:
         switch (wNotifyCode)
         {
           case MCI_NOTIFY_SUCCESSFUL:
              /* This indicates the command was completed successfully */
              break;
           case MCI_NOTIFY_SUPERSEDED:
              /* This indicates another notification request for   */
              /* the same type of command was received.           */
              break;
           case MCI_NOTIFY_ABORTED:
              /* This indicates that the command was interrupted   */
              /* and cannot be completed.                          */
              break;
           default:
              /* Notify code is error code.  You can use this      */
              /* error code with mciGetErrorString API.            */
              break;
     }
     break;
}
```

*Figure 6: Sending notification messages*

As an example, the user can start a Duet Player and play one of the duets, then go to another application and play a waveform file from that application. When this happens, MDM sends the Duet Player a message indicating that it has lost use of its devices. When the duet player receives a MM_MCIPASSDEVICE message from MDM indicating that it has temporarily lost control of the devices owned by the Duet Player, it sets a global flag to keep track of this fact.

Suppose a user wants to hear the rest of the duet. If he or she clicks on the Duet Player's window to activate it, a WM_ACTIVATE message is sent to the Duet Player's window procedure. This tells the Duet Player that the user wants it to be the active window. The Duet Player must regain control of any devices it may have passed since it was last activated. The Duet Player checks that global PassedDuet flag and reacquires its devices if it has passed them away, as shown in Figure 7.

Again, we use the mciSendCommand, this time with an MCI_ACQUIREDEVICE flag to indicate that we wish to get the devices back. MDM handles all processing required to restore the devices to their original state. If the device was playing a duet when we passed control of it to another application, upon reaquisition it will be playing the duet exactly where it left off when it was passed.

### *Multimedia Help*

The Duet Player programs also illustrate how to incorporate multimedia into an application's help information using OS/2's information presentation facility (IPF). A user can include an .IPF script tag in a help file to provide a link to additional information. In the case of the Duet Player programs, an audio help file is opened and the audio help is played.

The :LINK. tag in the .IPF file activates a link to additional information, as shown in Figure 8. The REFTYPE=INFORM tag causes an HM_INFORM message to be sent to the application when the user selects the "linked" area.

When a user selects "Play Audio Help" in the .IPF help panel, IPF sends an HM_INFORM message to the application's window procedure, as shown in Figure 9. When the application receives this message, it can take over and provide whatever multimedia function it wishes to add to its help.

When the Duet Player receives the HM_INFORM message, it calls AudioHelpWindowProc to play the audio help. Although the Duet Player is not required to implement audio help in a separate window procedure, this method simplifies the program for illustrative purposes and makes it easy for application developers to copy this portion of code into their own applications.

In the audio window procedure, an `mciSendCommand` function is issued to open the audio device using the audio help file and play the audio device setting the `MCI_NOTIFY` flag. When the file finishes playing, the MDM returns a notification message to the help window procedure and the device is closed. This procedure is very similar to that used to play the duets.

## STRING TEST

In contrast to the Duet Player, which illustrates the procedural interface into MMPM/2, the String Test program, shown in Figure 10, illustrates the interpretive string interface provided by the media control interface of MMPM/2. Using the String Test, you can enter string commands to control multimedia devices installed on the system. You can also view messages posted to the application as a result of entered string commands and select the types of notification messages to be displayed.

String Test is more than a sample program; it also serves as a powerful testing and debugging tool for developers writing media drivers. It lets developers control their own media devices interactively instead of having to write multiple test scripts.

The string interface has a few advantages over the procedural interface illustrated in the Duet Player samples. It lets users interactively control devices with a PM or command line interface. In addition, applications that use script languages can integrate string commands into those languages, allowing developers to integrate multimedia into their applications with little cost. Finally, in certain programming circumstances, the string interface is easier to use than the procedural interface. As in the Duet Player example, the procedural interface requires data structures and flags to be set up. With the string interface, however, you can simply pass a character string buffer to the API. Some developers prefer to manipulate strings instead of dealing with messages and data structures; others prefer the opposite. Choose the interface that best suits your preferences and requirements.

The application sends textual strings to the Media Control Interface using the `mciSendString` API. Figure 11 shows a sequence of string commands that open a wave audio device with an alias, load and play a wave audio file on the device, and finally close the device. Note the similarities between the string command and the message, data structures, and flags associated with the procedural interface illustrated in Duet Player I.

```
case WM_ACTIVATE:

      /* We use the WM_ACTIVATE message to participate in device
       * sharing.  We first check to see if this is an activate
       * or a deactivate message (indicated by mp1).  Then,
       * we check to see if we've passed control of the duets'
       * devices.  If these conditions are true, then we issue
       * an acquire device command to regain control of the
       * device, since we're now the active window on the screen. */

        if ((BOOL)mp1 && fPassedDuet == TRUE) {

           mciGenericParms.dwCallback =  (DWORD) hwnd;

           dwError = mciSendCommand( wGroupHandle,
                                     MCI_ACQUIREDEVICE,
                                    (DWORD)MCI_NOTIFY,
                                    (DWORD) &mciGenericParms,
                                    (WORD)NULL);
      }
```

*Figure 7: Using the* `WM_ACTIVATE` *message to participate in device sharing*

```
:link reftype=inform res=1.Play Audio Help:elink.
```

*Figure 8: Linking to additional information*

```
case HM_INFORM:
   WinSendMsg( hwndAudioHelp,
               UM_PLAY_AUDIO_HELP,
               (MPARAM) NULL,
               (MPARAM) NULL);
   return( (MRESULT) 0);
```

*Figure 9: Selecting play audio help*

### *Program Flow*

Figure 12 illustrates the interaction between the MMPM/2 system components and the String Test sample program.

1. String Test waits for you to enter a string command. When you select the "Send push" button, the program calls `mciSendString` to send the string command to the MDM.

2. The MDM receives the string command and uses a command table to parse the string and convert it to the command format of the media control interface. (Media devices can install customized command tables in MDM that allow it to understand string commands unique to the device.)

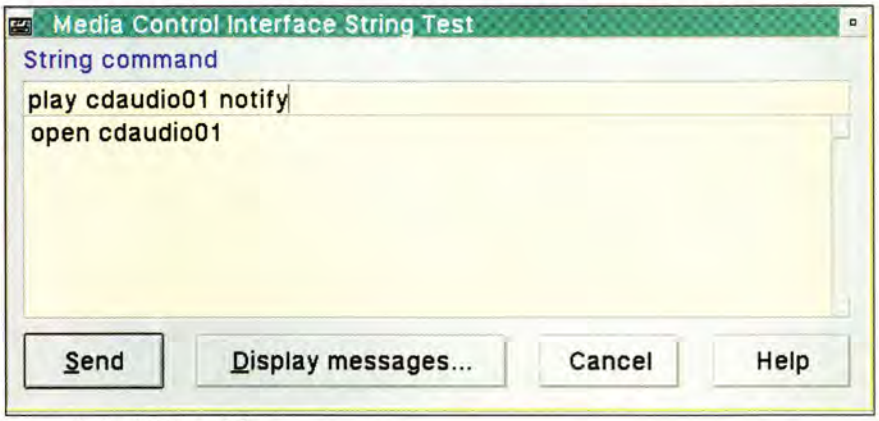



*Figure 10: Send String window*

```
/* Get the MCI String Command that is supposed to be in
   the Entry field of the combination box.            */

GetStringFromComboBox(
   hwndComboBox,
   &acMCIString[ 0 ] );

lSendStringRC =
   mciSendString(
      (LPSTR) &acMCIString[ 0 ],       /* The MCI String Command.   */
      (LPSTR)&acMCIReturnString[ 0 ],  /* Where to put Return strings */
      (WORD) MCI_RETURN_STRING_LENTGH, /* Size of the Return space  */
       hwndDisplayDialogBox,          /* Which window recvs. Notifies  */
       usCountOfMCIStringSent );       /* The user parameter.       */
```

*Figure 11: Sending string commands*

3. When the string has been parsed, MDM uses `mciSendCommand` function to send a command to the appropriate media driver, which acts on the command.

4. The media device sends notification messages to the application to indicate such operations as completion of a device function or the passing of device ownership from one process to another. These messages can be viewed in the Display Messages dialogue window of the String Test sample program.

## CLOCK SAMPLE

The Clock Sample, shown in Figure 13, is a simple analog clock that displays the system time. It creates a unique chime for each quarter hour, and, if the user requests, can display a visual indication of the chime.

The Clock Sample illustrates the use of MMPM/2's closed-captioning indicator to decide if it should provide users with a visual cue of its chimes. If the closed-captioning indicator is set, a chime causes an image of a bell to swing back and forth in the lower right hand corner of the window.

This sample also illustrates MMPM/2's memory playlist feature, which enables applications to manipulate multimedia data in memory to produce a variety of effects. With looping and branching instructions, the program creates audio chimes for the clock.

The `Multimedia Setup` program in the MMPM/2 folder allows users to turn a closed-captioning option on and off. Applications can issue the `mciQuerySysValue` function to determine if the user wants closed-captioning implemented in the applications on his or her system, as shown in Figure 14. Applications can check this flag and implement copies appropriately, as with the clock's swinging bell.

The memory playlist is a data structure in an application that contains an array of simple, machine-like instructions. With playlist instructions, you can play successive audio objects from one or more memory buffers. Looping and branching instructions are included in the memory playlist function. The application can also modify the playlist while it is being played. The playlist is a simple way to implement multimedia into games with MMPM/2, among other possibilities.

The clock uses the memory playlist to implement its chimes. It creates the chimes for each quarter hour using three short audio files in different combinations. As an alternative to recording the appropriate chime for every

quarter hour in the day, each chime is created dynamically based on the time. This memory playlist feature opens up many possibilities for dynamically created multimedia feedback. One practical application of this technique is in computer games.

## AVC I/O PROCEDURE INSTALLATION SAMPLE

The AVC I/O Procedure Installation Sample illustrates how an application can install and remove an I/O procedure to exploit multimedia I/O (MMIO) file services. The sample is a simple PM application that allows a user to install or remove the audio AVC I/O procedure AVCAPROC.DLL.

This sample shows the installation and removal of a system-defined I/O procedure. Typically, the user would install a customized I/O procedure not built into the `MMPMMMIO.INI` file during installation.

Similar to the media control interface subsystem, the MMIO subsystem is made up of two layers. When an application calls multimedia I/O functions, the MMIO Manager calls the appropriate I/O procedure, using it to direct the input and output associated with reading from and writing to different types of storage systems or file formats. I/O procedures provide an abstract of the file format that makes operations such as read, write, and seek independent of the specific format in use for applications. The handler is responsible for translating a generic application request into the necessary format-specific operations.

MMIO services provide four internal I/O procedures, as in Figure 15.

- *DOS*—Handles standard OS/2 disk files.
- *MEM (memory)*—Manages memory files without accessing the file system.
- *BND (bundle)*—Supports the elements in a RIFF compound file.
- *CF (compound file)*—Supports the RIFF compound-file format. The CF I/O procedure operates on the entire compound file rather than on the elements in a RIFF compound file (as does the BND I/O procedure).

The MMIO Manager can also install a custom I/O procedure to handle I/O to files of a format different from that of internal I/O procedures, such as AVC or M-Motion files. (MMPMTK/2 provides source code for a case-converter I/O procedure, which gives an example of what to consider when writing a file format I/O procedure.)

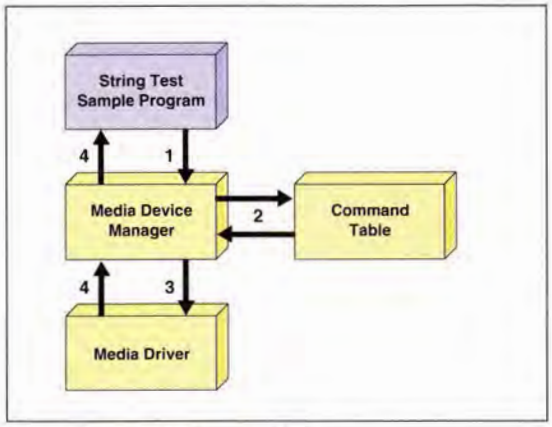



*Figure 12: String Test program flow*

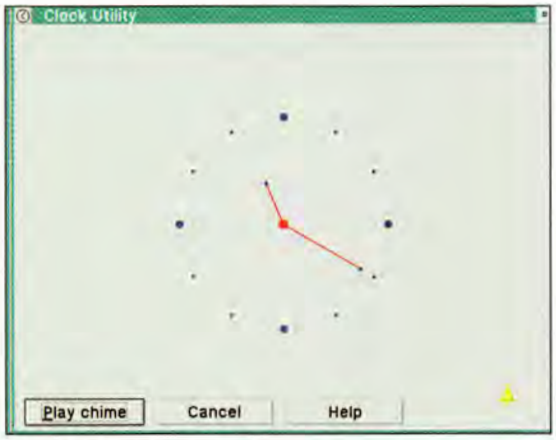



*Figure 13: Clock sample window*

The AVC I/O Procedure Installation program uses a semipermanent installation method that calls the I/O procedure only when a specific process is active. When the process ends, the I/O procedure is removed from its table. This is accomplished by calling the `mmioInstallIOProc` function during run time, as shown in Figure 16.

The first parameter, `fccIOProc`, is a four-character code, or FOURCC. A FOURCC is a 32-bit quantity representing a sequence of one to four ASCII characters—a unique identifier for a specific I/O procedure. Figure 17 explains how to obtain a FOURCC.

*The memory playlist feature opens up possibilities for dynamically created multimedia feedback.*

The second parameter in the procedure was obtained by issuing `DosLoadModule` and `DosQueryProcAddr` to the I/O procedure DLL.

```
mciQuerySysValue( MSV_CLOSEDCAPTION, (PVOID)&fClosedCaptionIsSet );
```

*Figure 14: Checking the closed captioning flag setting*

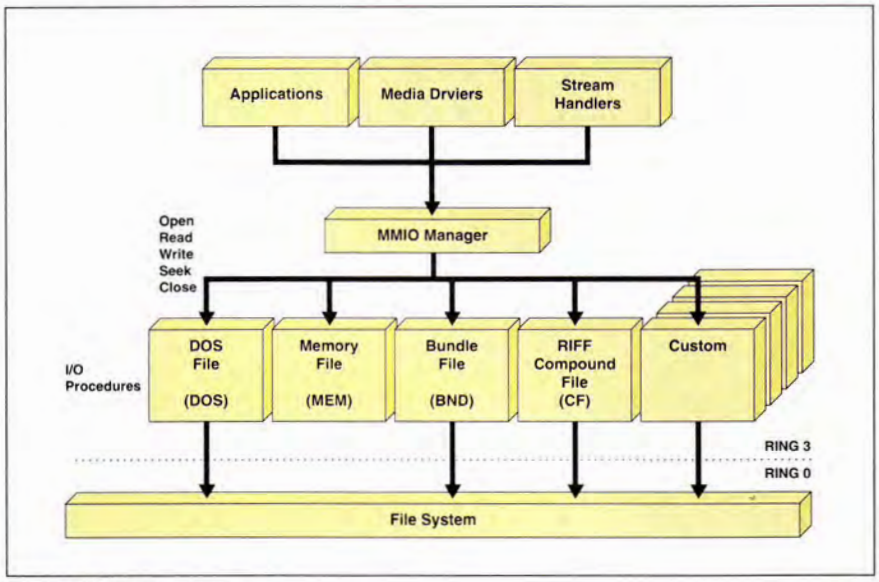



*Figure 15: MMIO subsystem*

```
pmmioprocSpecialIOProc =
    mmioInstallIOProc(
    fccIOProc,          /* Identifier of the procedure */
    lpmmioprocIoProc,   /* Pointer to the procedure    */
    MMIO_INSTALLPROC ); /* Flag to install the procedure  */
```

*Figure 16: Installing an I/O procedure*

```
FOURCC     fccIOProc    /* Four-character ID of IOProc.
fccIOProc = mmioFOURCC( 'A', 'V', 'C', 'A' ) ;
```

*Figure 17: Four-character code (FOURCC)*

## AVAILABILITY

MMPM/2 requires OS/2 2.0, and is available for $125. The MMPM Toolkit/2, which includes the base MMPM/2 product, is available on CD-ROM for $199. For more information about these products, contact your local IBM representative or IBM authorized remarketer, or call the IBM Multimedia Information Center at (800) 426-9402, xN80.

**Dana M. Liburdi**, *IBM Boca Raton Programming Center, 1000 N.W. 51st St., Boca Raton, Fla. 33429. Liburdi, a staff information developer, joined IBM in 1983 as a cooperative education student attending Florida Atlantic University. Her current responsibilities include planning and writing the MMPM/2 1.1 technical library.*

**Evi Larsen**, *IBM Boca Raton Programming Center, 1000 N.W. 51st St., Boca Raton, Fla. 33429. Larsen, a senior associate programmer, is currently the lead designer and developer of the MMPM/2 Toolkit. Prior to this assignment, she worked on the OS/2 1.2 and 1.3 Shell team, designing and developing the OS/2 user interface. Larsen has been with IBM since 1987, when she graduated from Florida Atlantic University with a B.A.S. in computer science.*

## REFERENCES

Noe, Brad, and Bill Lawton. "Introduction to IBM Multimedia Presentation Manager/2," *IBM Personal Systems Developer*. (Spring 1992): 72-77. (IBM Doc. G362-0001-13)

*MMPM/2 Programming Guide* (IBM Doc. S41G-2919)

*MMPM/2 Sample Programs Workbook* (IBM Doc. S41G-2921)

Multimedia

# Networked Full Motion Digital Video

*by Wendi Nusbickel and John Albee*

*This article discusses a networking full motion digital video with audio project, currently being tested in an OS/2 high-performance local area network (LAN) test lab in Boca Raton, Fla. The project is currently being presented at trade shows worldwide.*

## ORIGIN OF THE NETWORKED DIGITAL VIDEO PROJECT

At the Boca Raton project, researchers use digital full motion video with audio to demonstrate OS/2 and the IBM PS/2 servers and networks. IBM clients use the PS/2 ActionMedia II™ Display Adapter with digital video interactive (DVI). DVI allows users to play full motion video with audio at 30 frames per second (fps) across a 16 megabits per second (mbs) token-ring (T/R) LAN. At the spring 1992 IBM shows, 16 PS/2s running OS/2 2.0 were connected by two T/R LANs to a PS/2 server running OS/2 and LAN Server 2.0 Advanced. Each client played a different DVI file; all 16 ran simultaneously across the LANs at 30fps.

The DVI demonstrations aroused customer interest in IBM multimedia network products and led to the creation of a networked DVI test project in Boca Raton as well as high-performance LAN trade show demonstrations around the world. The efforts described here are intended as a general reference; another environment may yield different results.

## BACKGROUND

### *DVI Technology*

DVI technology is a combination of hardware and software that brings fully interactive multimedia applications—including full-speed, full-screen video, audio, still images, graphics, and text—to a PC. DVI can be used to access video, audio, and other program materials, from magnetic or optical computer disks to CD-ROM disks.

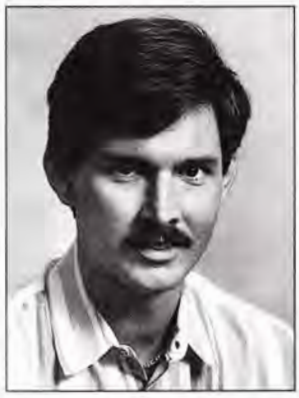
Unlike analog video products such as the IBM PS/2 TV (which uses analog video broadcast across T/R wires), DVI is a digital video product, stored in digital file format. Because a DVI file is stored on a PC disk, it can be transferred across a LAN like any user file.

### *ActionMedia*

Created in a joint project with Intel, the IBM ActionMedia 750 Delivery Adapter/A™ and the ActionMedia 750 Capture Adapter/A™ were the first commercial products to play and capture full-motion video with audio on a PC using DVI.

The IBM ActionMedia II Display Adapter with the capture option was a follow-up product with two times the processing power of previous ActionMedia products, improved video quality, single adapter implementation for display and capture, OS/2 support, scalable motion video, and improved real-time video compression. It was sold for 40% less than previous ActionMedia products.

*Wendi Nusbickel*



*John Albee*

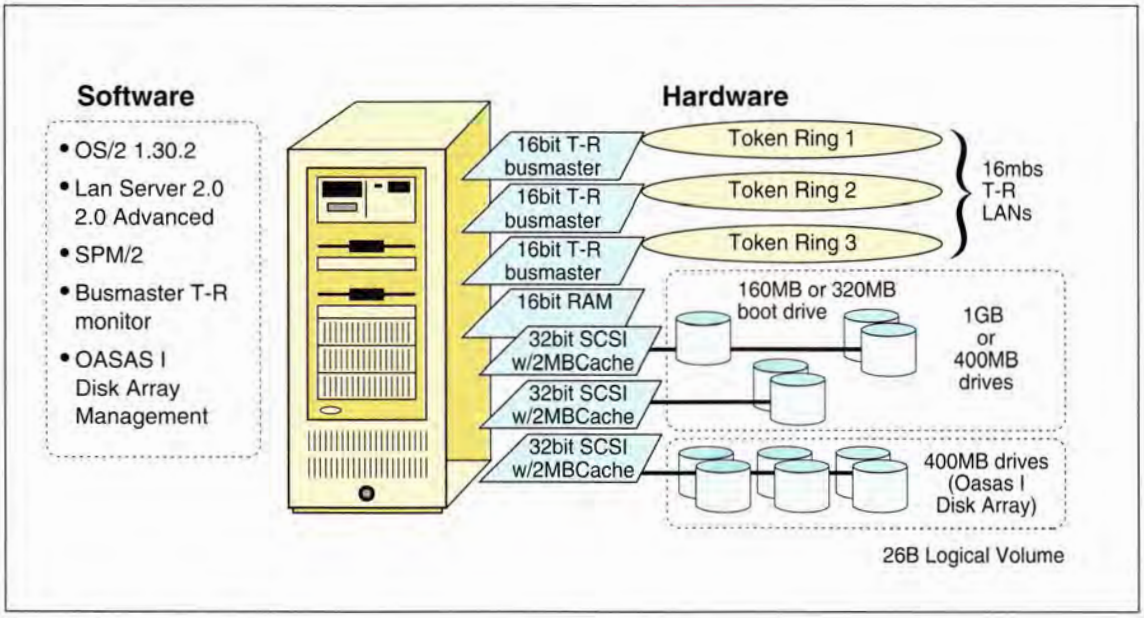



*Figure 1: Server configuration*

*DVI files requiring minimum decompression time and maximum video quality are good candidates for PLV compression.*

### *DVI File Compression*

DVI files are compressed to reduce the storage space and required delivery device speed.

Real-time video (RTV) is compression produced on the ActionMedia adapter in real time. With the capture option, analog video and audio is captured from a source such as a VCR, digitized, compressed in real time, and stored on the computer's fixed disk.

Production-level video (PLV) is high-quality compressed video produced by a licensed PLV compression facility. Compression is performed off-line, and takes advantage of asymmetric compression algorithms. These algorithms optimize decompression time from the ActionMedia adapter (at the expense of compression time) and are far more extensive than those used for RTV. DVI files requiring minimum decompression time and maximum video quality are good candidates for PLV compression.

### *LAN Technologies*

There are several ways to attach personal systems in a network. The two most popular ways used today are Ethernet and T/R LAN. These two LAN adapters, manufactured for most available PCs and mid-size systems, are widely supported by LAN software. Their popularity has been determined in part by their speed and cost. Ethernet supports speeds of up to 10mbs, while T/R supports up to 16mbs. Fiber distributing data interface (FDDI) surpasses both, supporting speeds of up to 100mbs. FDDI is not yet in common use, however; it is an emerging technology that is still relatively expensive.

## NETWORKED DIGITAL VIDEO

The per-system investment for DVI technology was previously quite high due to the necessity of having a CD-ROM and large disk drives to accomodate required files. Using a server to store these files eliminates the need for a large amount of disk storage space on each client. In fact, with all DVI files stored on the server, client systems can even be medialess remote initial program loaded (RIPL) clients.

Storing DVI files on a server allows many clients to share real-time DVI data on a LAN. A client can be set up to capture "live" data or images onto a server that can be accessed by other clients. This system opens the market for full-motion video with audio in customer industries such as videoconferencing, airlines, medical, travel, advertising, and education.

Uncentralized DVI files can quickly become out of date. File maintenance is simpler when all data is located on one server. Every client accessing DVI files from the server

automatically receives the most current data. A client also receives data contiguously across the LAN, which may sometimes be faster than accessing data locally. This is because DVI files on local media may become fragmented and file access from low-cost workstation media may be slower than file access from the LAN.

## DATA RATES, STORAGE, AND COMPRESSION

### *CD-ROM Delivery Rate*

A DVI file stored and played back from a CD-ROM at 30 fps, or full motion, has a delivery rate of 150K per second. An application playing a DVI file compressed at this rate expects data at 150K per second. The CD-ROM delivery rate is used as a compression guideline for networked DVI files. Each client on the LAN must therefore receive DVI data at a continuous rate of 150K per second or 1.2mbs to maintain full motion.

### *Data Storage*

With an average compression ratio of 150:1, a file that plays back at 150K per second requires 9MB of storage for every minute of play.

### *Network Delivery Compression Considerations*

Bytes-per-frame (BPF) refers to the maximum number of bytes that can be stored in a compressed video frame. Increasing the amount of data in a frame increases the network bandwidth, which affects network delivery. Depending on the amount of other networked trafficsupported by the LAN and server, drastic increases in the BPF can overload the network delivery components such as the server's resources and LAN itself. These components may be unable to support the increased data per second required for playback. At the CD-ROM delivery rate of 150K per second, the BPF is 5K (5K x 30fps equals150K per second). IBM's networked DVI environment uses a maximum BPF of 5K.

PLV target decompression time is an important factor in playing PLV compressed DVI files. Also referred to as decode or video decode number of field times, target decompression time is the time allowed to decode one video frame. Decompression time is expressed as a multiple of field times (one field time is approximately 1/60 second). A decompression time of two indicates that decompression is allowed to use all available playback time. Decompression times of less than two allow time for other processing, such as getting data from the network to the ActionMedia adapter. For PLV played across a LAN, IBM uses a target decompression time of 1.7.

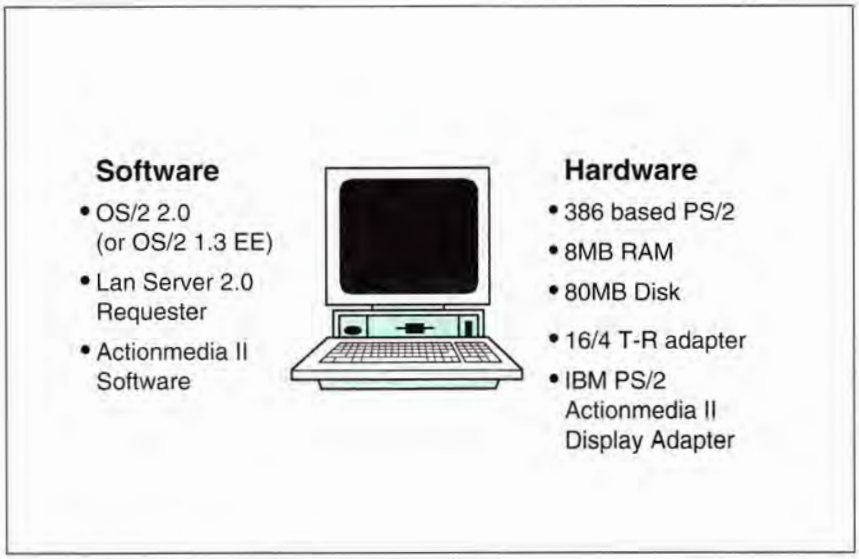



*Figure 2: Client configuration*

## COMPONENTS OF THE NETWORKED DVI ENVIRONMENT: SERVER PLATFORM

### *PS/2 Hardware*

The hardware platform for IBM's server started out with the PS/2 model 8595-0KD (33MHz, 486DX). As the PS/2 processor technology evolved, this platform changed; at the time of this article, the platform in use was the PS/2 8595-0MT model (announced in the spring of 1992), containing leading-edge 486DX-50MHz processor technology. Although this high-end PS/2 is not required for the networked DVI environment, it was designed to enhance overall LAN server function, reliability, and performance. It brings double the Micro Channel bandwidth (40MB

per second) of previous PS/2s and improved overall data integrity to the server platform.

*Other Storage Devices*

IBM tested a CD-ROM on the server, but found that due to the speed of the device (150K per second), it could support only one DVI playback client at a time. We also used an optical "jukebox," the IBM 3995 Optical Library Dataserver. In a "future technology" demonstration at COMDEX Fall 1991, we showed a high availability storage subsystem, a hardware SCSI hard disk array using redundant arrays of inexpensive and independent disks (RAID) level five fault tolerance, currently in development at IBM.

*With LAN Server 2.0 Advanced, no special code is required to enable DVI file service.*

*Software*

At the time of this article, the operating system on the server is OS/2 1.30.2, running the server software LAN Server 2.0 Advanced (a high-speed network server based on IBM LAN Server 1.3). For a fault-tolerant disk array, we use the OASAS I disk array management software for OS/2. Monitoring software is the IBM System Performance/2 (SPM/2) and a prototype T/R monitor. We are currently migrating our OS/2 server to use OS/2 2.0, a follow-on LAN server product, and a new high-speed hardware disk array.

*High Performance LAN Server 2.0 Advanced*

With LAN Server 2.0 Advanced, no special code is required to enable DVI file service. Its server code fully utilizes the architecture of the 386 and 486, including 32-bit instructions and data, to produce a fast, efficient server. Its OS/2 device drivers make good use of server hardware such as the busmaster T/R and SCSI adapters. In addition to these performance improvements, LAN Server 2.0 Advanced includes a number of functional enhancements such as support for up to four LAN adapters and the ability to support remote IPL OS/2 clients on the LAN.

*Key Components—High Performance Disk Interface*

HPFS386 is a 32-bit version of the high performance file system (HPFS) first implemented in OS/2 1.2. This server version of HPFS386 forms the core of the LAN Server 2.0 file service. A high-speed interface between the HPFS file server and the disk subsystem, designed to accelerate LAN I/O, provides the multirequest, asynchronous I/O necessary for high performance. As this interface is activated only for HPFS formatted volumes, all DVI files should be stored on HPFS formatted hard-disk drives and disk arrays.

*PS/2 Hardware Use*

A busmaster adapter relieves the CPU by using its own processor for some tasks, freeing the CPU for other tasks and allowing the busmaster to operate independently of the CPU. This improves system throughput and allows for better I/O management. The busmaster SCSI and T/R adapters also support the subsystem control block (SCB) architecture, which issues multiple requests in a list or command chain to the adapter. Previously, there was a single command interface between the CPU and device. Without the command chaining ability, the CPU would have to wait for each individual command to complete before issuing the next. Device drivers using this SCB architecture, such as the OS/2 LAN Server 2.0 busmaster T/R and SCSI disk device drivers, have a significant performance gain over conventional OS/2 device drivers.

*OASAS I Disk Array — High-Availability Disk Array*

The OASAS I Disk Array Management Software for OS/2 manages an array of SCSI hard-disk drives, providing data "striping," fault tolerance, and media spanning. Striping spreads data across all drives in the disk array. OASAS I uses RAID level five fault tolerance, which is striped data and error correction code (ECC) across all drives in the disk array. If any drive in the array fails, data can be accessed through an error-recovery procedure completely transparent to OS/2 and user applications.

*High Availability for the Multimedia Server*

In IBM's networked DVI environment, many DVI files are stored on an OASAS I disk array attached to the server. Clients on the T/R LAN can play DVI files from the disk array at full speed. If one of the disk array drives is turned off, the files play with only a slight degradation in performance.

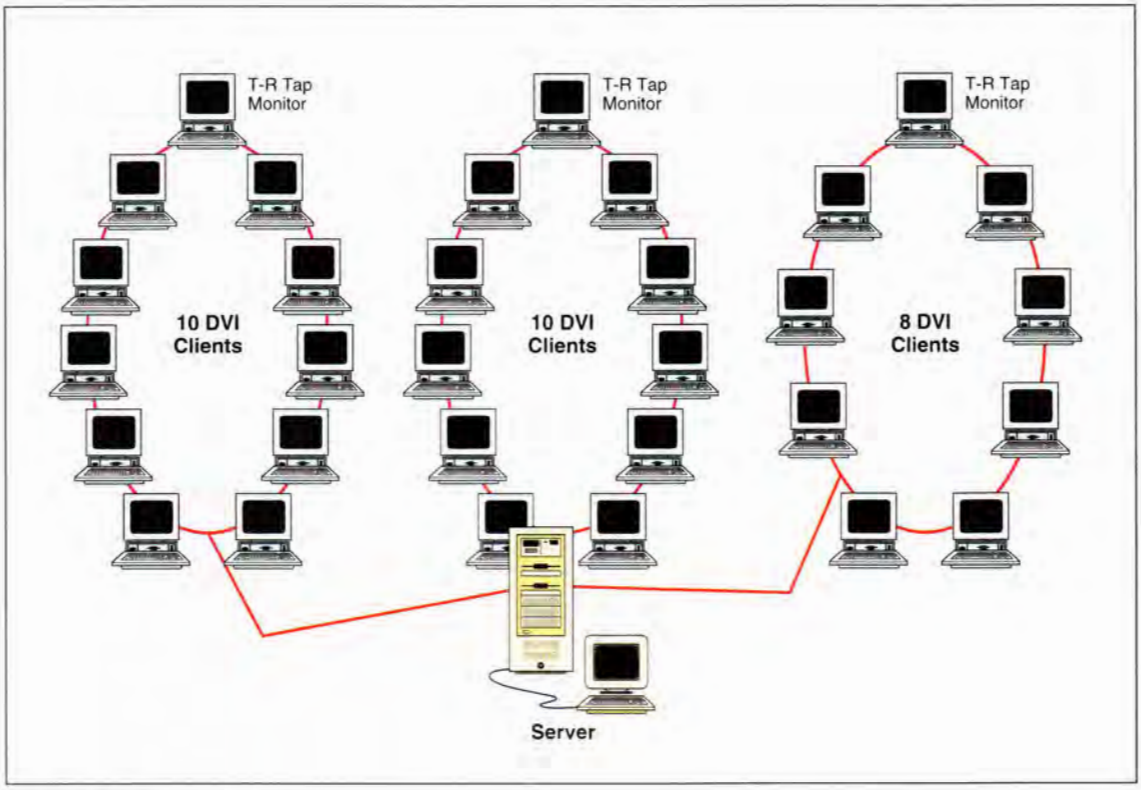



*Figure 3: LAN configuration*

### *Netware Server*

The initial project did not include a Netware server. Netware servers are now being used with DVI clients in areas such as education, and we are planning to test with one.

## COMPONENTS OF THE NETWORKED DVI ENVIRONMENT: CLIENT PLATFORM

### *PS/2 Hardware*

We change three of the hardware configuration defaults with hardware diagnostics on the clients. The ActionMedia display adapter extended memory window buffer is increased from 8K to 32K. The T/R buffer size (in the options menu, "RAM size and address range") is increased if it is less than 16KB. Finally, we lower the priority of the T/R interrupt level to avoid possible ActionMedia starvation. Since the ActionMedia Display adapter with the capture option defaults to use interrupts 9, 10, and 11, we change the T/R adapter's interrupt level to three to lower the priority, because interrupts 8 through 15 all share interrupt level 2. (The lower the interrupt level number, the higher the priority.)

### *Software*

Our test lab clients run OS/2 2.0 with LAN Server 2.0 requester services. OS/2 Extended Edition 1.3 may also be used for the client platform. Installed on the clients are the ActionMedia II Audio Video Kernel (AVK) and toolkit software. The toolkit video with audio player program (splayer.exe), modified to support remote control messages from a single control client and report DVI fps, is used to play DVI files across the LAN.

## COMPONENTS OF THE NETWORKED DVI ENVIRONMENT: LAN

T/R was chosen as the initial LAN for this project, as it was the fastest, most affordable LAN available that does not use a protocol

dependent on collision sensing or avoidance. The T/R protocol can be seen as a train, with the token as locomotive and data requests and data as cars. Data is added to the token at the server and removed at the client making the request, allowing equal client access to the data server. If a T/R is overloaded, it slows client access to data, as it may not be able to add requests until room opens up. In contrast, with a collision-sensing LAN like Ethernet, if a network overloads, all traffic stops.

*The T/R protocol can be seen as a train, with the token as locomotive and data requests and data as cars.*

Figure 3 shows the IBM networking DVI test environment, containing three 16mbs T/R LANs supporting 28 simultaneous full motion DVI PS/2 clients from a single server. The final goal of this project is to configure four T/R LANs with 40 clients from our server.

## COMPONENTS OF THE NETWORKED DVI ENVIRONMENT: MEASUREMENT TOOLS

### *IBM System Performance Monitor/2 (SPM/2)*

SPM/2 is an OS/2 software package that monitors system performance and can be used to analyze performance problems. It is commonly used as an aid for performance tuning and load balancing. In this project, the SPM/2 real-time monitor facility was used to analyze and demonstrate resource utilization within the server. Its real-time monitoring capability allowed immediate interpretation of system performance at our trade show demonstrations. The SPM/2 monitor displays a graph of the CPU, disk, and RAM usage in the system, and can be used to identify system bottlenecks.

The latest SPM/2 customer service diskette also supports tracing of LAN Server 2.0 Advanced HPFS formatted volumes and drives managed by the OS/2 OASAS I disk array software.

### *Disk Load Balance*

SPM/2 is used primarily to balance workload between the hard disk drives. In its sampling period of between five and 60 seconds, SPM/2 determines what percentage of a resource is in use; a resource accessed throughout the entire sampling period has a utilization of 100%. The SPM/2 monitor displays a separate colored line on the disk graph for each physical disk in the system, which indicates which drives are being under- or over-utilized. This method also determines the load placed on each SCSI adapter.

### *IBM T/R Network 16/4 Trace and Performance Program (TAP)*

TAP is a multipurpose utility that provides T/R monitoring and analysis. It runs in a client installed with DOS 3.3 or above, attached to IBM's T/R network 16/4 Trace and Performance Adapter/A. TAP shows a sliding bar graph indicating the percentages of the T/R used for user data and for data and any T/R overhead (such as for tokens).

### *Busmaster T/R Monitor Prototype*

With the busmaster T/R device driver OS/2 source code as a base, prototype code was written to call the OS/2 system trace hooks. This code traces the amount of data transferred within the busmaster T/R device driver on the server. The system trace hooks are picked up by a prototype OS/2 Presentation Manager (PM) monitor program, similar to the SPM/2 monitor, also running on the OS/2 server. The PM monitor shows a graph of the composite throughput of the busmaster T/R adapters in the system, in either the megabytes or megabits per second.

## HOW WE "NETWORKED" DVI

Networking a DVI application is just like networking any user application. The fact that the result is full motion video is transparent to the client DVI application and file server. To grant access to the DVI data and programs on the server, we `NET SHARE` the subdirectory that contains the DVI files on the server. Then we `NET USE` this resource at the clients. The drive redirection is transparent to the DVI application at the client.

## PERFORMANCE

The areas critical to the performance of this project are the disk drives, disk adapters, and T/R. Project results show that 10 clients can

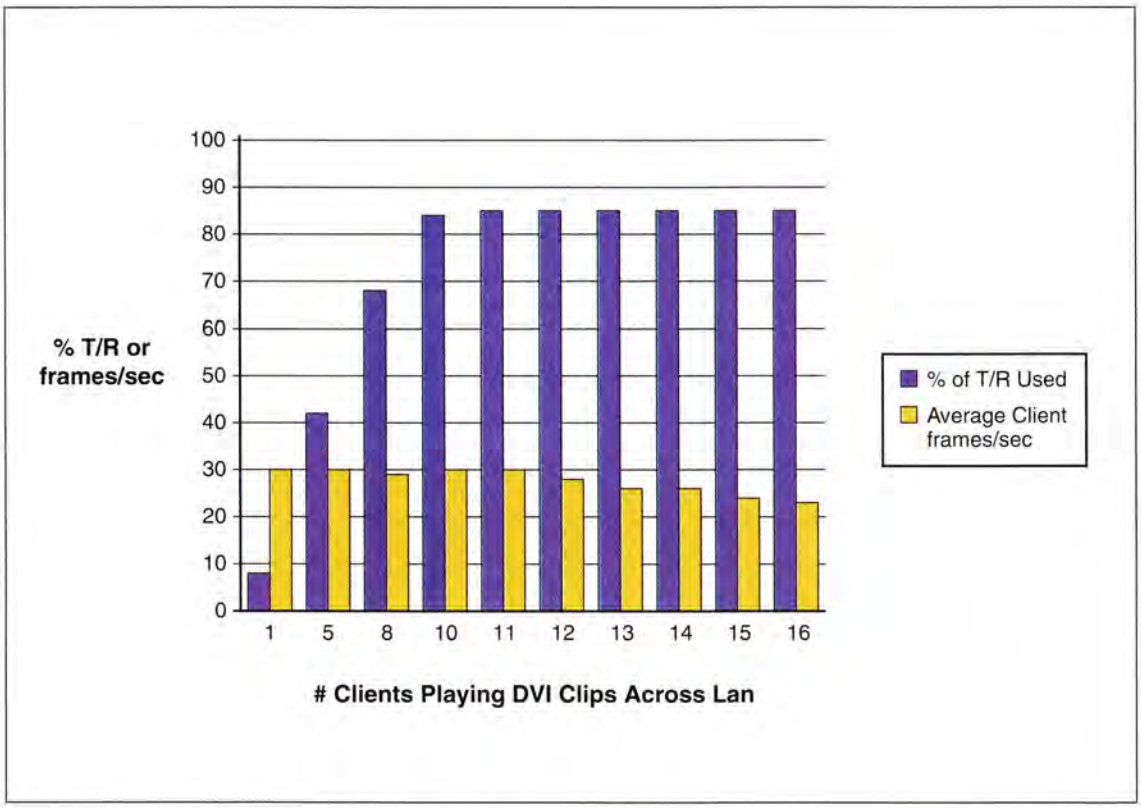



*Figure 4: Performance of DVI clients On single T/R LAN*

simultaneously play full motion video with audio on a single 16mbs T/R LAN. More than 10 clients results in degraded performance, which initially shows up as pauses in the audio tracks and jittery video. In addition, a single disk drive and controller could not simultaneously support all our networked DVI file accesses in the server. We therefore had to duplicate our DVI files across multiple disk drives and controllers and balance the client accesses accordingly, to avoid any possible disk drive intervention.

Figure 4 illustrates client performance degradation as 16mbs T/R utilization increases to the point of saturation of the busmaster T/R adapter. The percentage of the T/R used was based on the peak reading of the TAP monitor for the T/R data and overhead. The degradation in the DVI clients was measured by the average fps of DVI play. Note the decreased average fps as more DVI clients were added to a single T/R LAN. While more than 10 clients playing DVI at once could be supported under the right conditions, we recommend a maximum of 10.

***Results: T/R,***

*DVI Test Criteria*—In our networked DVI environment, we try to achieve as close to 30fps as possible on all clients. Audio breaks up at around 26fps; if the fps drops to less than 27 or jerkiness appears, we consider the test a failure. During the tests documented in this article, only DVI traffic occurred on the LAN.

*Maximum DVI Clients on a Single T/R*—Assuming the 1.2mbs rate required for a client to play DVI at a CD-ROM rate, our 16mbs T/R LAN supports 10 clients playing DVI simultaneously.

*Typical T/R utilization for 10 DVI Clients*—The T/R TAP monitor typically shows between 75% and 85% utilization on a 16mbs T/R with 10 clients playing DVI at once. Assuming a playback rate of 1.2mbs per client, for 10 clients the amount of data (not including LAN overhead) required is 12mbs, or 75% of a 16mbs T/R LAN. (The amount varies depending on how the DVI file was compressed.)

*While more than 10 clients playing DVI at once could be supported under the right conditions, we recommend a maximum of 10.*

*IBM Busmaster T/R Adapter Transfer Rate*—In our test lab, we attained a maximum speed of around 13.7mbs for the IBM busmaster T/R adapter.

*Supporting More Than 10 Simultaneous DVI Clients*—We could not support more than 10 clients playing DVI simultaneously because of the limited bandwidth of the busmaster T/R adapter in the server and the T/R LAN. To avoid this limitation, either use a LAN with higher bandwidth, such as FDDI, or install more than one T/R LAN. Because FDDI was not available at the time of this writing, we set up multiple T/R adapters in the server.

*Networked multimedia is in need of new supporting technologies.*

*Other Network Traffic*—There is nothing that distinguishes DVI data from any other data transferred over the network. The performance of DVI plays while other network traffic is occurring depends on the amount of network bandwidth available and on factors such as disk contention on the server.

***Results: Disk Subsystem***

*32-bit IBM SCSI adapter with cache*—The 32-bit SCSI adapter has an 8-bit data path and transfer rate on the SCSI bus of 5MB per second and comes with a 512K cache. However, because our networked DVI environment is so disk intensive, we increase the standard 512K cache to 2MB, using two industry standard 1MB-by-nine-DRAM SIMMs. When disks will be simultaneously accessed for numerous DVI files, we attach up to four SCSI disk drives to each SCSI adapter. Limiting the number of disk drives per SCSI adapter may only be necessary if the amount of data required per second exceeds the data transfer rate of a single adapter.

*IBM 1GB and 400MB SCSI Hard-Disk Drives*—The 1GB SCSI fixed disk drives have an 11.0 millisecond average seek time and a 256K look-ahead buffer. Using these drives on the server, we are able to play five different DVI files simultaneously from each disk drive. The 400MB SCSI fixed disk drives have an 11.5 millisecond average seek time with an 128K look-ahead buffer. With these drives on the server, we are able to play three different DVI files at once from each drive.

***Results: CPU***

*CPU Utilization*—The CPU utilization in our server is usually around 25%. We attribute this low utilization to the fact that the server is I/O vs. CPU bound, and the adapters in the server are busmasters.

## FUTURE CONSIDERATIONS

Networked multimedia is in need of new supporting technologies. Some emerging technologies are currently available, but high prices prohibit their widespread use except in specialized applications. Improvements in the LAN transportation medium (wire, coax, and so on) and protocols are necessary to reduce the number of physical rings necessary to support a large number of clients. Faster disk subsystems that can support more concurrent multimedia file accesses are also needed to eliminate multimedia file duplication and load management.

FDDI provides considerably greater bandwidth than T/R, in theory allowing eight times the number of clients of a single T/R. FDDI, however, requires a relatively expensive adapter card for each system and requires that some current LAN cables be replaced.

We envision a need for protocols to support the synchronization of data and to reserve data capacity. Data synchronization would allow multimedia clients to receive data from multiple sources and present that data concurrently. For example, if the video and audio are not in the same digitized file but are presented concurrently at the client, the LAN would have to insure that both streams of data arrive at the client simultaneously.

Due to the limited bandwith capacity of today's LANs, it is not possible to guarantee every user full-motion video with audio without physically limiting the number of client workstations. This is impractical in a typical LAN environment with hundreds of users using the LAN to access conventional data such as documents or spreadsheets. If it were possible to reserve a percentage of the bandwidth for multimedia use, adequate performance could be ensured to a finite

number of multimedia users. Bandwith reservation is valuable with both 16 and 100 megabit per second (mps) LANs.

Larger and faster storage devices and their supporting subsystems are necessary to keep up with the growing needs and varieties of multimedia data. They also reduce the need to insure adequate user access by replicating data across different drives and controllers.

Finally, increasing the current compression ratio of multimedia data without loss of quality would relieve some storage and network speed and capacity requirements.

## CONCLUSION

The ability to network full motion video on a T/R LAN can strategically aid IBM's high-end PS/2 server and OS/2 2.0 clients by broadening the multimedia market to include areas such as education and advertising, with centralized multimedia data and reduced workstation investment.

According to the computer trade press, networking may be the force that makes DVI technology succeed. "Proponents of digital video interactive (DVI) have said the multimedia-derived technology is waiting for a spark to set off rapid growth. That spark [is] networking," wrote Jim Nash in *Computerworld* magazine. "DVI...is already having an impact on employee training and product marketing. The problem is, it is often impractical to set up an entire system—including a PC and videocassette or laser disc player—on an employee's desk. The answer...is to centralize the data and distribute it over local area networks." (Nash, et al.,1991)

Networked DVI and multimedia can be achieved with available IBM products. But as multimedia networks become more prevalent, the need for increased and guaranteed network bandwidth and larger, faster, and more fault-tolerant storage devices will become crucial.

**Wendi I. Nusbickel,** *IBM Personal Systems Line of Business Division, 1000 N.W. 51st St., Boca Raton, Fla. 33431. Nusbickel is an advisory programmer on the OS/2 Servers and Special Projects team. She joined IBM Boca Raton in 1983, where she worked on the IBM PC XENIX™ project. She was in OS/2 development for four years before becoming a member of the OS/2 Servers department. She has a B.S. in computer science from North Dakota State University in Fargo, North Dakota.*

**John P. Albee,** *IBM Personal Systems Line of Business Division, 1000 N.W. 51st St., Boca Raton, Fla. 33431. Albee joined IBM in 1979, working with mainframe communications until the advent of the IBM PC. He supported PC communication products through the IBM technical coordinator program and has been involved in system integration work and workstation strategy. His last assignment was with IDS strategy and architecture. Albee is currently a senior technical planner with the OS/2 Servers and Special Projects team in Boca Raton, Florida. He holds a B.A. from the University of Connecticut.*

## REFERENCES

Nash, Jim. "Networking could spur DVI technology growth," *Computerworld*, April 22,1991: 49.

Multimedia

# MMPM/2 and OS/2: The Multimedia Advantage

*John E. Parsons*

*Val Enright*

*Maria Ingold*

*by John E. Parsons and Val Enright*

*The effective use of sound, image, animation, text, and video typically imposes requirements that are well beyond anything ever imagined in the design of PC operating systems. Unlike text editors or other applications that can gracefully wait until another task has yielded use of the processor, multimedia applications often cannot tolerate processing delays without significant loss in the quality of the presentation. A good example of this is audio breakup, which occurs when an application cannot send digitized audio to an audio adapter at the required rate.*

*One approach to solving the problems caused by processing delays would be to let a multimedia application retain exclusive use of the processor. But this approach would conflict with another multimedia requirement: responsiveness. A multimedia operating system must provide superior and seemingly dedicated throughput while at the same time remain responsive to interactive applications. This can only be accomplished by an operating system that maximizes allocation of its resources using such key technologies as preemptive multitasking, priority scheduling, overlapped I/O, and demand-paged virtual memory. OS/2 2.0 uses these technologies and delivers an ideal multimedia environment.*

*The first few sections in the article briefly describe OS/2 features with an emphasis on the particular benefit to multimedia applications in the Multimedia Presentation Manager/2 (MMPM/2) environment. The rest of the article describes MMPM/2 features and functions.*

*References are made in the article to the publications in the MMPM/2 Technical Library. A list of these publications with their descriptions, order numbers, and prices can be found at the end of this article.*

## MULTITASKING

Multitasking describes the ability of an operating system to give the appearance of processing more than one task (or program) concurrently. Although several methods of multitasking are available, two predominant methods include cooperative and preemptive multitasking. Cooperative multitasking gives the appearance of concurrent processing by allowing applications to give up or yield the use of the processor when it is no longer needed, for example, yielding the processor while waiting for user input.

Cooperative multitasking can be highly effective for conventional applications, but its premise that all software developers will cooperate in adhering to a set of rules on how the processor should be used is too simplistic. If any developer deviates from the rules, either accidentally or intentionally, the system rapidly degrades to a single-tasking environment. The deterministic throughput requirements of many multimedia applications make this environment unacceptable.

Preemptive multitasking implements a scheme whereby the operating system is in charge of effectively allocating the valuable resources of the processor. Each application that requires the processor is transparently dealt a time slice, giving each application the impression that it has exclusive use of the processor. Those applications waiting for user input or the completion of other events will not use a time slice. Not only does this scheme greatly simplify development of application software, it also maximizes the use of the processor. Starvation of any one application is prevented

because the operating system shares the processor among all applications.

Because OS/2 2.0 uses preemptive multitasking, MMPM/2 can effectively provide multimedia capabilities to applications while still maintaining a very responsive system. Applications can initiate the playback of audio data and let the operating system schedule the processor so that audio breakup does not occur.

Multitasking is much more than running multiple programs at a time, however. It provides a developer with tools to write more modular code that exhibits better performance and responsiveness to the user. An OS/2 developer can cause parts of a program to be spun off into separate units of execution, called threads. With proper design, it is often possible for different parts of a program to be running simultaneously.

For example, if a user selects an option to save a file, a separate thread can be initiated to actually save the file, enabling the application to return immediately to the window that has the input focus. Thus, the application always remains responsive to the user instead of locking up the user interface until the operation has completed. MMPM/2 uses the advanced multithreaded design of OS/2 2.0 to achieve high levels of concurrency, while presenting a simple programming interface.

## PROTECTED MEMORY MODEL

All versions of OS/2 have provided a protected memory model that allows the operating system to run applications in separate address spaces. This model effectively prevents any one application from corrupting the memory of another. Its primary benefit is the prevention of system failures that require restarting the system because of an ill-behaved application. System failures caused by errant code can be frustrating to users. The resultant loss in productivity is magnified several times if the user is also a software developer who spends a significant amount of time debugging code.

An ill-behaved application running on OS/2 2.0 meets with abrupt and consistent termination whenever it attempts to corrupt the system, enabling users to immediately resume their work, rather than wait until the system is restarted. Because MMPM/2 is implemented on the same solid foundation as OS/2 2.0, applications using the features of MMPM/2 automatically exhibit the same data integrity.

### *Paged Virtual Memory*

In addition to the protected memory model that has been part of OS/2 since its inception, OS/2 2.0 provides access to memory in 32-bit linear addresses. Unlike its predecessors,

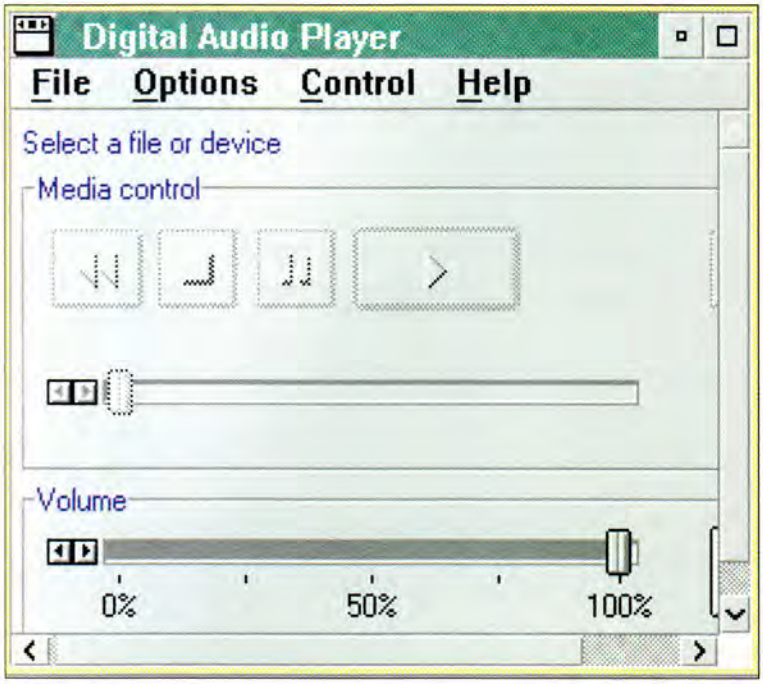



*Figure 1: Digital audio player*

which used selector and offset addressing, OS/2 2.0 uses data segments that are not limited to 64KB in size. Removal of the 64KB limit has helped to reduce the extra effort incurred by requirements to manage large amounts of data. This enhancement is especially important in multimedia applications, which can have very large data objects present in memory. For example, one second of compact disc quality audio requires 172KB of storage.

Another enhancement specific to OS/2 2.0 is the use of paged memory. Unlike the segmented architecture of earlier versions of OS/2, which swapped programs and data in and out of memory in chunks up to 64KB in size, OS/2 2.0 divides a program and its data into manageable and uniform 4KB pieces.

This addressing scheme prevents a problem that can occur in segmented architectures when memory segments are of different sizes. Gaps can occur in allocated memory that, when added up, are large enough to meet a specific memory request. However, because the unallocated memory cannot be addressed by a single selector, it cannot be used without the costly overhead of memory compaction and its resulting performance impact.

OS/2 2.0 implements a paging scheme that presents memory as a nonsegmented space up to 512MB.

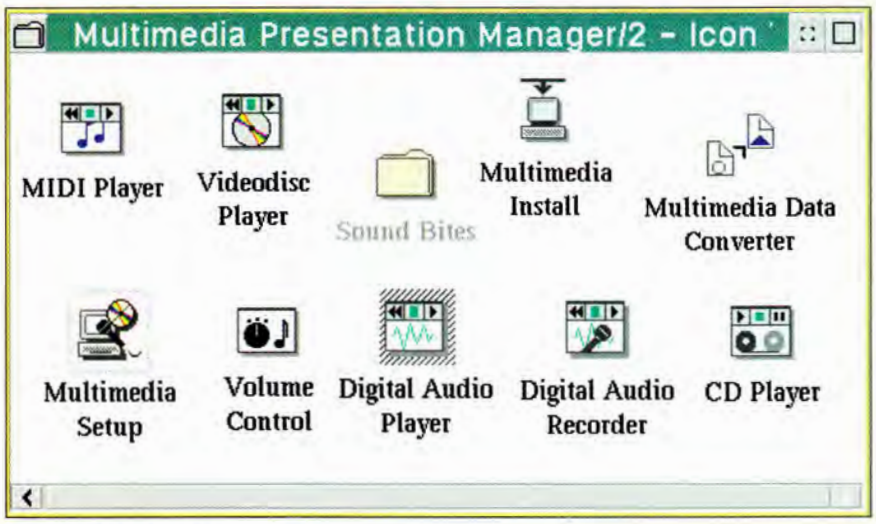



*Figure 2: MMPM/2 folder*

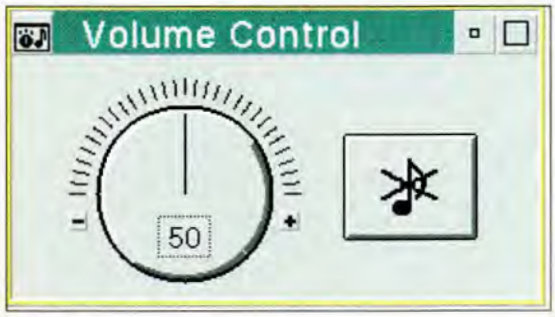



*Figure 3: Volume control application*

## ADVANCED USER INTERFACE

An advanced object-oriented human interface complements the technological powerhouse provided by the operating-system kernel. OS/2 2.0 implements the workplace model of the Common User Access™ (CUA) 1991 specification, referred to as the Workplace Shell. Productivity gains resulting from increased usability are significant not only for the end user, but also for the software developer. OS/2 2.0 provides several advanced user interface controls including notebooks, sliders, and containers.

Notebooks resemble their physical counterparts in that they are organized into sections divided by tabs. The user can leaf through the notebook a page at a time or use the tabs to move quickly from section to section. The tabs can contain text or graphics to communicate their meaning. The MMPM/2 Multimedia Setup application uses a notebook to maintain the settings and user preferences associated with each multimedia device.

Sliders are similar to scroll bars in that they provide a simple mechanism that enables the user to quickly move across a wide range of values. The subtle difference between a scroll bar and a slider is illustrated by the use of a scroll bar to move the viewport of a text editor to a specific location within a document, whereas a slider is used to set the value of a particular item. The MMPM/2 media players use sliders to control the volume and media position of multimedia devices in the system. The Digital Audio Player is shown in Figure 1.

Containers are controls that can contain other system objects. A good example of containers are the folders used in the Workplace Shell. Figure 2 shows the Multimedia Presentation Manager/2 folder, which contains several useful applications and sample data files. The samples, which include digital audio and musical instrument digital interface (MIDI) selections, can be played on the MMPM/2 media players. There are no restrictions on the use of these samples.

Although this example illustrates the use of containers as a graphical equivalent of a conventional file system directory, containers can be used for any abstract collection of objects.

## MULTIMEDIA WINDOW CLASSES

MMPM/2 defines several new Presentation Manager window classes that lend a sophisticated multimedia quality to a user interface. These controls are the graphical button, circular slider, and the secondary window, as illustrated in Figure 3 by the

MMPM/2 Volume Control application. The Volume Control application provides system-wide control of the master volume level of all active multimedia devices. This allows the user a single point of interaction for volume control.

Graphical button controls allow replacement of conventional text-faced buttons with either two-state or animated graphics. Standard text-faced buttons can be replaced by graphical buttons without any change in source code. Only the dialogue template needs to be modified. Implementing two-state and animated buttons requires a minimal investment in development time equivalent to implementation using the standard push-button control.

Circular slider controls present an interesting alternative to the Workplace Shell slider control in that the physical appearance of the control is very similar to the controls on actual multimedia hardware. This resemblance allows users to transfer knowledge acquired from using real devices to multimedia applications. The circular slider can also provide a savings in screen real estate compared to a conventional slider. The programming effort required to implement a circular slider is equivalent to that required to implement a standard slider or scroll bar.

Secondary window controls represent a significant savings in development expense because all application windows can be developed using a dialogue template, eliminating the need to create, position, and size windows and controls. All of the MMPM/2 system applications use secondary windows. If the window is sized smaller than the actual dialogue template, scroll bars can be automatically enabled to allow access to the entire dialogue.

For more information on the advanced multimedia controls provided by MMPM/2, see the *MMPM/2 Programming Guide* or *MMPM/2 Programming Reference*. For information on design concepts to be considered when designing a CUA multimedia interface consistent within a particular application and across other products, see the *CUA Guide to Multimedia User Interface Design*.

## MULTIMEDIA LOGICAL DEVICES

MMPM/2 provides OS/2 2.0 applications with device-independent control of an entirely new class of devices that support multimedia function. Support for devices such as audio adapters, CD-ROM drives, and videodisc

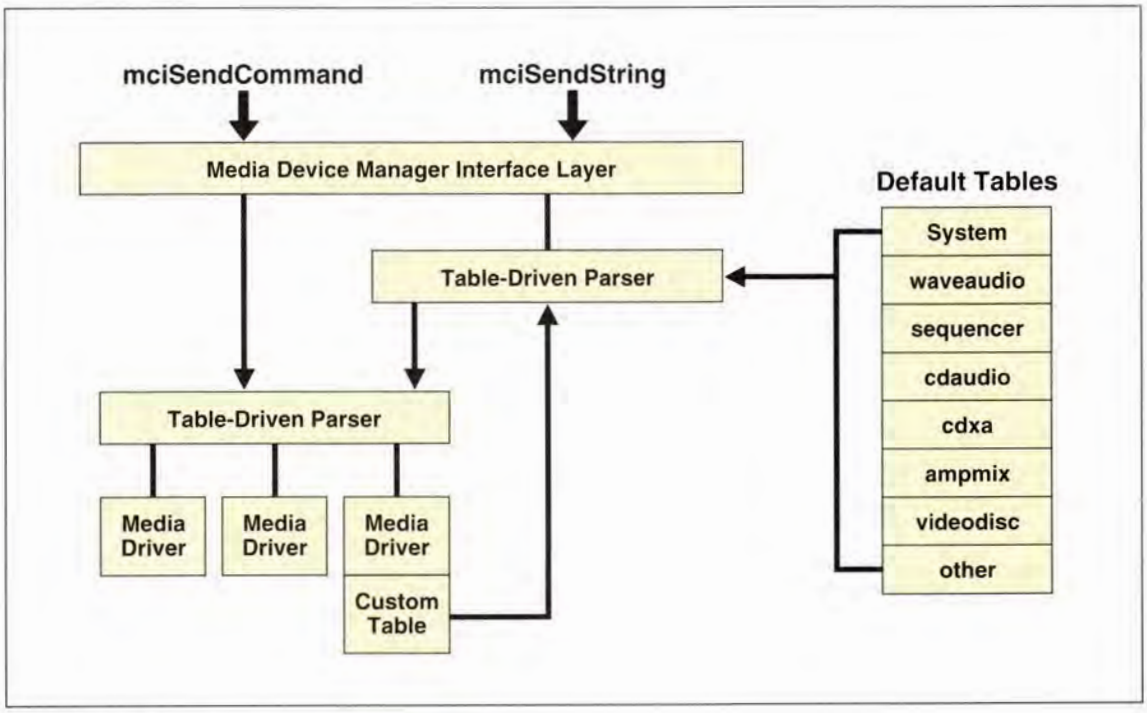



*Figure 4: Media Device Manager and MMPM/2 devices*

players is provided through the mechanism of logical media devices managed by the Media Device Manager™ (MDM), shown in Figure 4.

In addition to providing the framework into which additional media drivers may be installed, MDM allows sharing of the actual hardware by multiple applications. MDM will exploit any sharable features of the hardware; if the hardware's resources are exceeded, MDM can temporarily suspend a background application's use of the device and allocate resources to a foreground application. Applications are notified of this type of sharing; however, the details of saving and restoring device states are transparent to the application.

*In addition to providing the framework into which additional media drivers may be installed, MDM allows sharing of the actual hardware by multiple applications.*

```
open cdaudio01 alias cdaud1 shareable wait
status cdaud1 media present wait
status cdaud1 mode wait
set cdaud1 time format milliseconds wait
seek cdaud1 to start
play cdaud1 notify
*** Time Elapses ***
close cdaud1
```

*Figure 5: String commands to play a CD*

There is frequently a one-to-one correspondence between a hardware device, such as a CD-ROM drive and its associated logical media device. In other cases hardware is represented as multiple logical devices. An example of the latter is a multifunction audio adapter, which is represented as waveform audio, MIDI sequencer, and amplifier-mixer media devices. Logical media devices currently supported by MMPM/2 include:

- Amplifier-mixer
- Waveform audio
- MIDI Sequencer
- CD audio
- CD-XA
- Videodisc
- Digital video.

Because MMPM/2 is extensible, additional media devices can be supported as they become available. Actual implementation of MMPM/2 device support is not relevant to an application because MMPM/2 provides device independence with the command message and command string interfaces.

When a user activates an application button or a slider to use a device function, the application window procedure responds by sending a command to the Media Device Manager (MDM). Depending on the needs of the application, the window procedure can use a command message interface or a command string interface to implement these device commands. The command message interface requires the usual C programming constructs, which include messages and pointers to data structures, whereas the command string interface requires only strings, which are sent to the MDM for parsing. For example, an application could send a series of string commands such as the ones in Figure 5 to open a CD player and play an entire CD.

Authoring languages that include support for the Media Control Interface can integrate device command strings with authoring language syntax to create multimedia presentations. In addition to the native 32-bit interface, the string interface can also provide a 16-bit interface to enable developers to integrate multimedia function with the macro languages of existing 16-bit applications. Following are brief descriptions of the logical devices currently supported by MMPM/2. For more detailed information on MMPM/2 devices, see the *MMPM/2 Programming Guide.*

### *Amplifier-Mixer Device*

The MMPM/2 amplifier-mixer device is very similar in function to a home stereo amplifier-mixer. Components are plugged into the amplifier-mixer so that audio signals can be transferred to a pair of attached speakers, headphones, or perhaps another device. One example of transferring audio signals to another device would be to play an old phonograph record on one device and record its sound on a new digital audio tape (DAT) deck. The amplifier-mixer serves as the central focus for all audio signals and provides input or output switching as well as sound-shaping services such as volume, treble, or bass control.

The logical amplifier-mixer device in MMPM/2 supports the connection of analog and digital devices. Other MMPM/2 logical devices may be connected to the amplifier-mixer device. Similar to the previous example, the CD audio logical device could provide an analog input to the amplifier-mixer device, which could then be recorded by the waveform audio device.

### *Waveform Audio Device*

The MMPM/2 waveform audio device allows an application to play or record digital audio using files or application memory buffers. While "audio" refers to the sound waves that have a perceived effect on the human ear, "waveform" refers to a digital representation of the original audio sound wave. Using one technique called pulse code modulation (PCM), discrete samples of the sound wave are encoded by an audio adapter at precise intervals. The numerical value of the sample increases when the sound wave's force (loudness) increases. The variation of the sample increases as the frequency of the sound wave increases.

The number of samples per second taken of the original sound wave as well as the precision (or resolution) of the sample dictate the quality of the sound reproduction. Typical sampling rates include 44.1kHz, 22.05kHz, and 11.025kHz, where kHz is an abbreviation for kilohertz or thousands of cycles per second. The sampling precision is usually measured in bits where 8- or 16-bit samples are representative of most audio adapters. Generally, the higher the sampling rate and resolution, the higher the perceived quality; however this comes at the expense of potentially enormous data rates and file sizes.

One of the typical uses of the waveform audio device is to digitize an input signal or sound into discrete samples for storage in a file. An example of this would be recording an electronic audio mail message to actually tell someone about an idea, as opposed to typing a memo on the same subject. An electronic audio mail application would be completely shielded from the complexity of digitizing a signal and would only need to specify a file, while providing the user with a simple control panel to record the message. The user might press a `STOP` button on the control panel when finished describing the idea. The application could then issue a `STOP` command to the waveform audio device to discontinue the recording.

### *Sequencer Device*

General MIDI (Musical Instrument Digital Interface) is a standard specification for playing back music from a series of commands, rather than actual audio data. The commands represent musical events, such as turning a note on and off as well as timing mechanisms for specifying the duration of the note sound.

The MMPM/2 sequencer device plays a MIDI song by sending commands from a MIDI file to a synthesizer, where the commands are converted to the sounds of a specific instrument. The sequencer uses the timing commands to sequence the playing of the music. Typically, a digital signal processor (DSP) is used to generate the sounds of the instrument, which results in an authentic reproduction of the original performance.

*A digital signal processor is used to generate the sounds of the instrument, resulting in an authentic reproduction of the original performance.*

MIDI augments waveform audio as a means of producing sounds in the multimedia environment. MIDI data offers the advantage of requiring far less storage than waveform data. For example, suppose a three-note chord, middle-C, E and G, is held for one second. The storage required to store this information as 16-bit, PCM, 44.1kHz, stereo waveform audio data requires 172KB, while the storage required for the MIDI commands is only 18 bytes.

Another advantage of storing musical performances as a series of instructions is that the information can be edited the same way words in a document can be edited by a word processor. The musical editing process can be used, for example, to correct mistakes in an artist's original interpretation or change certain points of style before playback or final recording. Playback of MIDI data using the sequencer media device can be used to reproduce the original performance.

### *CD Audio Device*

The CD audio media device provides access to devices that read compact discs for the purpose of playing CD audio. A typical use for CD audio is to provide high-quality audio for use in a presentation. Another use of CD audio

would be to provide detailed audio help for an application user. Instead of the usual hyperlinked text and graphics, an entire step by step audio tutorial might be stored on a compact disc in several different languages.

A mixed-format compact disc holds CD-ROM (compact disc read-only memory) file system data as well as CD-DA (compact disc digital audio) track format data. An example of the use of a mixed format disc is an application that contains several symphonies by a famous composer. The actual audio is stored as a series of CD-DA tracks. Also stored on the disc (but in CD-ROM file format) is a program, the

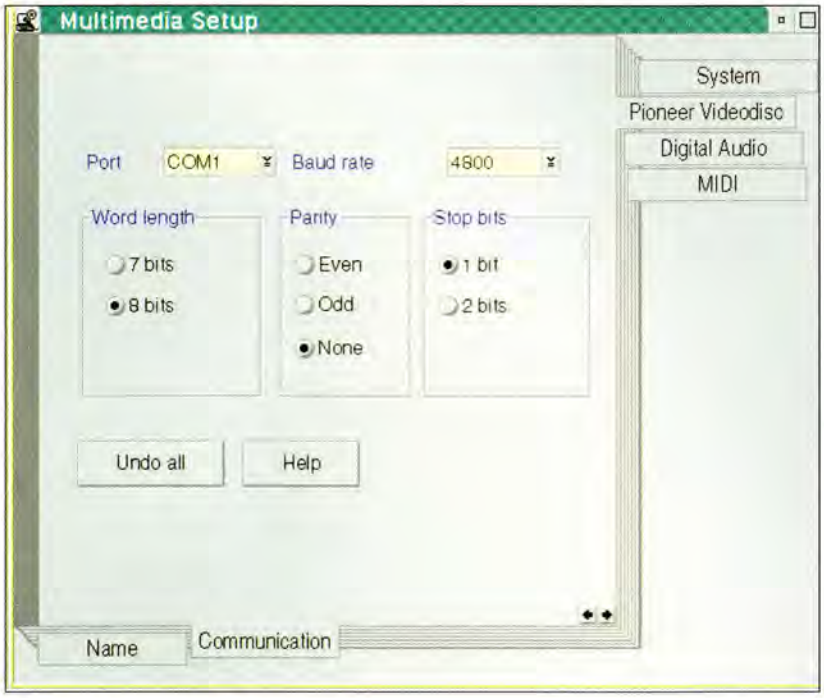



*Figure 6: Videodisc notebook page*

actual music score, and perhaps a database on the composer's life. When the application is started, the audio from a symphony can be played using the CD audio media driver, while the user is allowed to study the music score. Additionally, the user might retrieve facts on the composer, such as how old the composer was when the symphony was written.

Depending on the type of CD-ROM drive installed, the audio data on a CD-DA disc is either processed by a digital-to-analog converter (DAC) built into the drive, or it is moved through the system to a DSP on an audio adapter. Some CD-ROM drives can only play CD-DA audio data through the built-in DAC. Others, like the IBM PS/2 CD-ROM-II Drive, can play through the DAC, or they can stream data through the audio adapter DSP.

The advantage offered by playing CD-DA through the DAC is that it is a simple operation that greatly reduces system and resource overhead. The advantage gained by streaming data through an audio adapter DSP is that you can potentially enhance the signal beyond the capabilities of a built-in DAC by adding special effects with a full-feature DSP—capabilities the built-in DAC may not provide.

### *CD-XA Device*

The CD-XA media device provides access to devices that support CD-ROM/XA discs. CD-XA (compact disc extended architecture) refers to a storage format that accommodates interleaved storage of audio, video, and standard file-system data. CD-XA data is stored in a file-system format on the discs, and playback control is managed by the CD-XA media device in cooperation with the amplifier-mixer device. The digital audio data is reproduced by an audio adapter, while the video and data segments are delivered to the application for appropriate processing or display.

CD-XA takes advantage of a special adaptive delta pulse code modulation (ADPCM) audio compression, which not only yields a low data rate but also enables more audio data to be stored on a disc than that allowed on a CD-DA disc. The ADPCM audio compression technique allows up to a 16-to-1 compression of audio data.

By compressing the audio data (in some cases to 1/16 the size of CD-DA data) it now becomes possible to record multiple audio tracks on a single disc. With CD-XA level C recorded in stereo, it is possible to interleave eight different audio tracks on a single disc. With CD-XA level C recorded in mono, this number climbs to 16 different tracks on a single disc.

### *Videodisc Device*

The videodisc media device provides control of videodisc external hardware devices that play videodiscs, producing an analog video and audio signal. The analog video output can be connected to an analog video digitizer component that produces output on the

display, or it can simply be connected to an external video monitor. The audio output can be connected to amplified speakers, headphones, or other audio media drivers.

The IBM videodisc media device supports Pioneer™ models 4200, 4300D, 4400, and 8000. The 4300D model supports the American (NTSC) and European (PAL) TV standards for storing analog video signals.

*Digital Video Device*

The MMPM/2 digital video device can lend an application the power of full-motion video. Typically accompanied by one or more synchronized audio tracks, digital video can be displayed on a computer screen, using a variety of software techniques and hardware assists. As with all MMPM/2 logical devices, applications are shielded from the details of implementation.

MMPM/2 currently supports ActionMedia II implementation of a digital video device. The ActionMedia II media driver, packaged with multimedia adapters, is easily installed. Additional digital video device support will be provided in MMPM/2 as media drivers become available. (For more information on ActionMedia II digital video support, see "ActionMedia II Support for MMPM/2," pages 111-116 of this issue.)

## INSTALLATION ENGINE

Delivered with every copy of MMPM/2 is an easy-to-use, graphical multimedia installation program. Building on key features of the OS/2 2.0 installation program, Multimedia Install makes installation simple, often requiring only one mouse click. What is unique about MMPM/2 Install is that it can be used as an installation engine. Text-based control files are created to describe the components of a product, which appear as iconic objects within a container control. The objects can be selected as a whole or individually if the user wants to pick and choose. Attached to each object is a list of required files and CONFIG.SYS updates. Because users will already be familiar with multimedia installation, this program serves as a natural way to deliver new multimedia products. See the *MMPM/2 Programming Guide* and the *MMPM/2 Sample Programs Workbook* for more details.

## MULTIMEDIA SETUP

The Multimedia Setup application uses the new OS/2 2.0 notebook control to provide users with a convenient method for maintaining the settings for multimedia devices. The settings for each device are presented as a page in a notebook. MMPM/2 provides settings pages for each media device to allow the name of the device to be customized and allow data files to be associated with the device. In addition to the standard settings, custom pages can be inserted into the notebook to exploit unique hardware requirements. An example of this feature is the custom page created for a videodisc device, as shown in Figure 6.

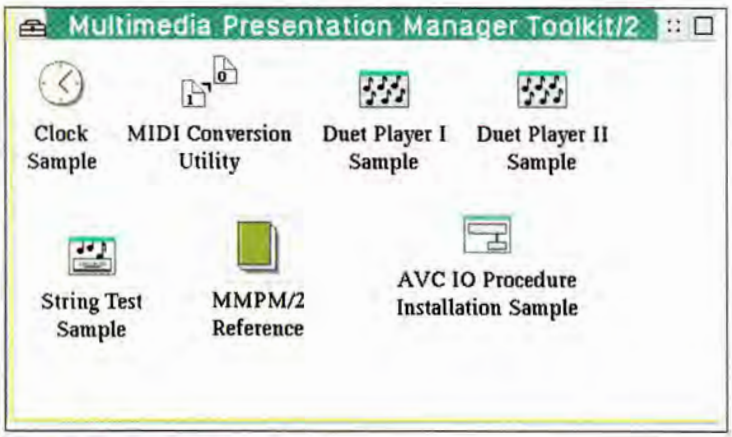



*Figure 7: MMPM/2 Toolkit folder*

Using the metaphor of a notebook, all multimedia device settings are controlled from a central and familiar place, greatly enhancing the usability of the system. While multimedia technology and hardware devices can be complicated, the use and configuration of these devices need be no different than any other device in OS/2 2.0.

## MULTIMEDIA PRESENTATION MANAGER TOOLKIT/2

Product developers will be able to increase their productivity through MMPM/2 and the MMPM/2 toolkit. This starts with a simple installation from CD-ROM using the multimedia installation program.

Once installed, the toolkit sample programs illustrate the use of the comprehensive multimedia device and data handling

## EYE-CATCHING ARTWORK

*by Maria Ingold*

The box cover art for MMPM/2 embodies a unique approach toward product visualization and marketing. It achieves consistency and familiarity by effectively capturing buyer attention as well as conveying sufficient information about the product. The artwork incites further interest by utilizing state-of-the-art three-dimensional graphics technology, a concise design, and a visual representation of the product's functionality.

Digital audio support is depicted as a sine-like wave, which is transformed into the curved, rainbow-colored music staff that moves the viewer into the picture. MIDI is represented in its icon form by musical notes resting appropriately on the music staff. CD-DA, CD-ROM/XA, and videodisc support are represented by their medium, the disc.

The disc alludes to the audio aspect of multimedia by being the focal point for the musical notes and bars. Additionally, it represents the video portion and videodisc technology by being the iris of an eye. The video conceptualization is further realized by a filmstrip winding through the image whose individual frames depict the MMPM/2 logo, which appears in the product as the desktop folder icon and in the installation progress indicator.

I composed the image through a series of sketches that were created and rendered using a three-dimensional graphics package called Wavefront™ running on an IBM RISC System/6000™ workstation.

capabilities of MMPM/2. There are working sample programs for every major function in the MMPM/2 system. As the sample code can be freely incorporated into real products, developers can be immediately productive without incurring a steep learning curve. Product developers will be able to focus on product-specific function instead of investing large amounts of time providing the multimedia enhancements. See the article on MMPM/2 Sample Programs in this issue.

Complete on-line documentation provides a quick effective reference to all system APIs, messages, and features, including code examples and helpful notes.

**John E. Parsons,** *IBM Corp., Personal Systems Programming Division, P.O. Box 1328, Internal Zip 1508, Boca Raton, Fla. 33429. Parsons is an advisory programmer in the Multimedia Software Design department and was the lead designer for the initial release of MMPM/2. He has been with IBM for two years and is currently the project lead for the video extensions to MMPM/2. Parsons received a BS and MS degree in computer science from the Georgia Institute of Technology where he is also a qualified Ph.D. candidate. Previous to IBM, Parsons*

*worked for Harris Corp. as a lead software engineer in the development of advanced communications and teleconferencing systems.*

**Val Enright**, *IBM Corp., Personal Systems Programming Division, P.O. Box 1328, Internal Zip 1606 Boca Raton, Fla. 33429. Enright is a staff information developer in Information Development and has been with IBM for 12 years. Her responsibilities include technical writing of OS/2 programming information. Her current assignment is the video extensions to MMPM/2.*

**Maria Ingold**, *IBM Corp., Personal Systems Programming Division, P.O. Box 1328, Internal Zip 2200, Boca Raton, Fla. 33429. Ingold is a programmer in Multimedia Software Development. She is the owner of the MMPM/2 Multimedia Data Converter "applet," and functions as the resident art consultant. She received her B.S. in computer science, with a studio art minor from the University of New Mexico. Part of her studies were done at the University of Essex, U.K.*

## REFERENCES

Multimedia Presentation Manager Toolkit/2 publications can be ordered by calling 1-800-IBM-PCTB. Following are the toolkit publication prices and order numbers:

*MMPM/2 Programming Guide* —Describes application and subsystem programming interfaces to help you select and implement functions for OS/2 multimedia applications. $23.50 (IBM Doc. S41G-2919).

*MMPM/2 Programming Reference* —Provides detailed information on multimedia functions, messages, and data structures to enable you to write code for multimedia applications. $22.00 (IBM Doc. S41G-2920).

*MMPM/2 Sample Programs Workbook* —Gives code examples from MMPM/2 sample application programs and templates for subsystem components. $20.50 (IBM Doc. S41G-2921).

*CUA Guide to Multimedia User Interface Design*—Describes design concepts to consider when designing a CUA multimedia interface. $9.70 (IBM Doc. S41G-2922).

*OS/2 Multimedia Advantages* —Describes advantages offered by OS/2 and MMPM/2 multimedia platform. $3.70 (IBM Doc. S41G-2923).

Complete MMPM/2 Technical Library —$78.75 (IBM Doc. S41G-3321).

**NDP Fortran, C|C++ and Pascal Compilers**

- **VMS, VS and MS extensions**
- **Compatible with Framework and C Set — Fortran can call C Set or NDP C**
- **Uses IBM's Link386 to generate native OS/2 executables**
- **Can directly call the OS/2 APIs**
- **Generate globally optimized 32-bit mainframe quality code**
- **Support for all coprocessors and the i860**
- **Available for DOS, OS/2, UNIX and NT**
- **3rd party numerics and GUI Class libraries available**

**For more information, call our Tech Support Group at (508) 746-7341.**

**Microway**

**Research Park**
**P.O. Box 79**
**Kingston, MA 02364**

# Free OS/2 Utility Disk*

OS/2 Monthly Magazine, **the independent guide to OS/2 computing**. Whether you are New user, a Developer, or a even Windows-OS/2 user, OS/2 Monthly has invaluable information Info. Programming and User Articles, Reviews, Market info, all by OS/2 Experts. With a paid subscription you'll receive the disk and 1 year (12 issue) subscription. (*$10 value.) Visa, MC, Check, Money order accepted.

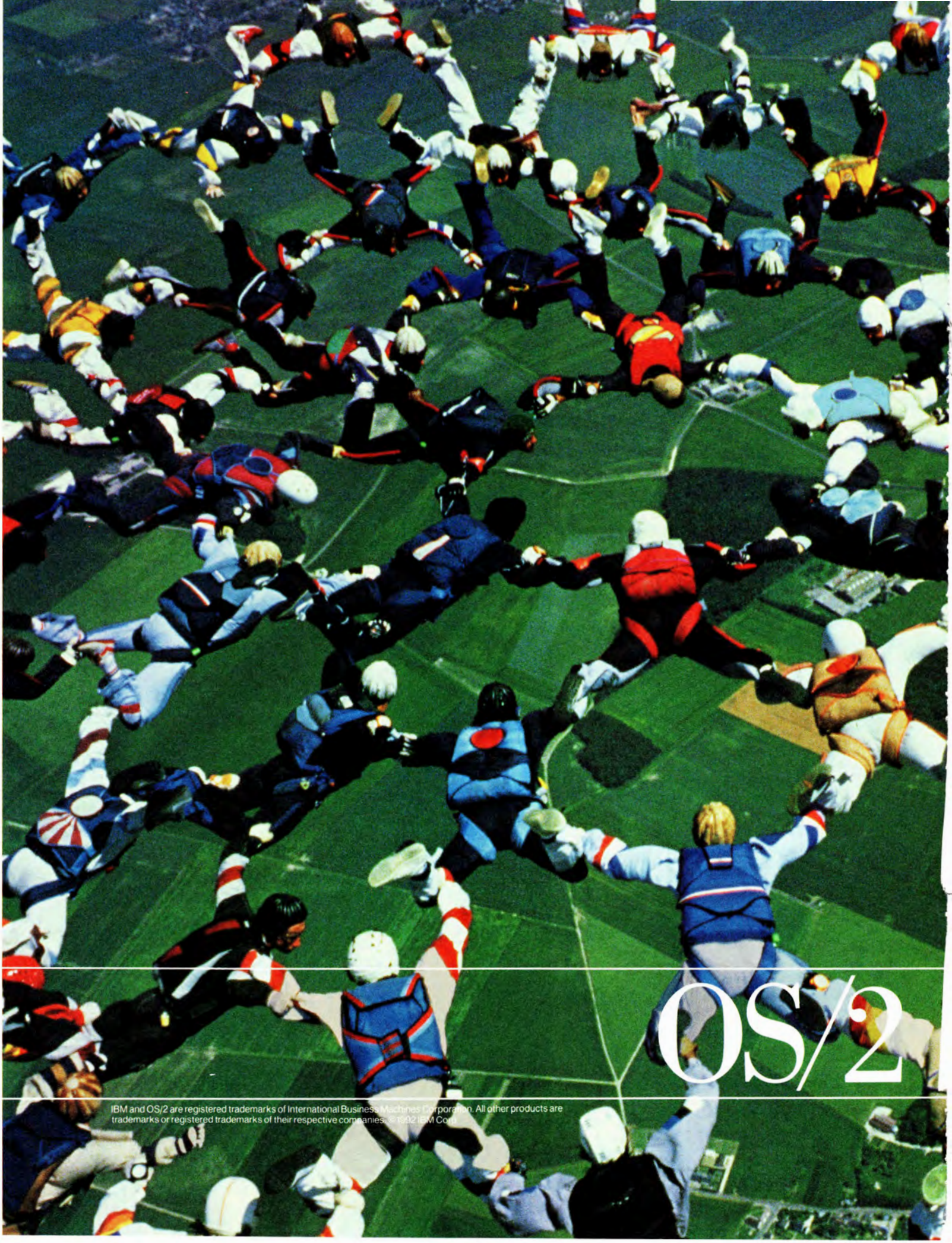
( ) **Bill me and start my 1 yr subscription (12 issues)**, when I pay the $39, I'll get the OS/2 utility disk.

Name
Company
Address
City State Zip
Credit Card
Expiry Date

Mail to: JDS Publishing, PO Box 4351, Highland Pk, NJ, 08904

**Addt'l $3 Discount for User Groups members or for mentioning the OS/2 Developer**

**For FASTEST service call 800-365-2642**




IBM and OS/2 are registered trademarks of International Business Machines Corporation. All other products are trademarks or registered trademarks of their respective companies. © 1992 IBM Corp.

# A lot can depend on how you're hooked up.

It's simple.

Sharing information and resources has never been more critical. That's why you need IBM LAN Systems. Because nobody else can offer the one clear overview that IBM can.

IBM LAN Systems has a comprehensive solution that will satisfy your specific needs, whether you choose our OS/2® LAN Server or NetWare® from IBM. Whether your hardware is IBM or not. And whether you run on DOS, DOS with Windows™ or OS/2. IBM LAN Systems supports all industry-standard networking protocols, including Ethernet and Token-Ring, as well as over 4,000 server-enabled applications.

Whether you use OS/2 2.0 as a server platform or as a client, with IBM LAN Systems you'll get not only seamless integration, but also a high-performance environment that makes the most of your LAN—with multitasking that lets more data travel at once, protected by fault tolerance and security functions. So even if you go down, you're not out.

Best of all, you get IBM's unparalleled service and support. So look before you leap into a LAN. You'll feel better taking the plunge with IBM.

## Introducing IBM LAN Systems.

- *Provides a comprehensive LAN solution.*
- *Choose between OS/2 LAN Server and NetWare solutions.*
- *Supports industry-standard protocols, under DOS, Windows and OS/2.*
- *Provides seamless integration and a high-performance environment.*

IBM®

Multimedia

# Multimedia Waveform Audio Support

John E. Parsons

Val Enright

*by John E. Parsons and Val Enright*

*Multimedia Presentation Manager/2 (MMPM/2) represents audio adapters, CD-ROM drives, videodiscs, and other hardware devices as logical media devices managed by the Media Device Manager (MDM). Frequently, there is a one-to-one correspondence between a hardware device, such as a CD-ROM drive, and its associated logical media device. In other cases, hardware is represented as multiple logical devices. An example of the latter is a multifunction audio adapter, which is represented as waveform audio, a MIDI sequencer, and amplifier-mixer media devices.*

*In this article, we describe the function and typical use of the amplifier-mixer and the waveform-audio media devices. We describe the software models these logical devices provide to the application developer, their relationship to the hardware, and how digital audio information flows into and out of an audio adapter using the software concept of connectors.*

## CONNECTORS: SOFTWARE CONDUITS OF MULTIMEDIA INFORMATION

A connector is a software representation of the physical way in which multimedia data moves from one device to another. A simple example is the headphone jack on a CD-ROM player or the speakers jack on an audio adapter. If an audio card had a speaker and line out jack, it would be desirable to let an application choose the destination of the audio, while at the same time remain independent from the hardware implementation.

MMPM/2 connectors provide this function by enabling an application to query which connectors are supported by a logical device and to manipulate the flow of information through the connector. The connectors for a logical device can be accessed either by number or a symbolic connector type. When specifying a symbolic type such as "microphone" or "line in," a number can also be specified to select the first connector, second connector, and so on, of that specific connector type.

Although connectors are typically associated with the representation of externally visible audio and video jacks on multimedia equipment, another category of connectors can represent the flow of information within a computer. One example includes a connector on an audio adapter that can be attached to the internal PC speaker. A more subtle example would be the flow of digital audio information into an audio adapter. This information could come from a file, system memory, or another device. Connectors of this category are typically referred to as "stream" connectors to convey the idea of a logical stream of information flowing from one device to another.

### *Using the IBM M-Audio Adapter*

Figure 1 illustrates how the capabilities of an audio card might be modeled as an MMPM/2 amplifier-mixer device using connectors. This example uses the IBM M-Audio Capture and Playback Adapter™; however, a model can easily be defined for any manufacturer's audio card. Only the number and type of connectors may vary.

In this particular model, the speakers(1) connector is the default speakers connector and represents the physical speaker jack on the M-Audio card. The speakers(2) connector represents the internal connection to the PC speaker. The `amp stream` connector represents the flow of digital information to and from the audio card.

### *Connections between Devices*

A connection is defined as the establishment of an information flow from one device connector to a compatible connector on another device. One example of a connection is the attachment of a speaker to an audio card with a piece of speaker wire. Typically, an application might enable the speaker connector on an audio card, causing the flow of information out of the speaker connector. If you have connected the wire, then the audio will be heard. In this case, MMPM/2 must rely on a person to make the actual connection.

Another category of connection exists where the connection is made internally in the computer, typically through the transfer of digital information from one media device to another. For instance, the waveform audio media device has a connection to its associated amplifier mixer device.

### *Default and Device Context Connections*

Device connections are usually automatically established by the media device when the device is opened. The choice of connection is determined by a default that was established during the installation of the media driver and can be reestablished using the `MCI_DEFAULT_CONNECTION` message.

Once opened, the media device may open and connect to another media device to provide the complete function of the originally opened device to the application. This is transparent to the calling application. One example is the waveform audio device, which uses a connected amplifier-mixer device to actually produce sound from the digital audio stream. The waveform audio device also uses the services of the amplifier-mixer device to set the volume.

While some of the services of the connected device can be surfaced in the definition of the originally opened device, the connected device can also provide some extended features beyond those required by the original device. If the application wishes to access these extended features, it can get the handle to the particular device context or instance of the connected device, using the `MCI_CONNECTION` message.

There is a subtle difference between a default and device-context connection. A default connection is the internal name of a connected device, whereas a device-context connection is the actual handle to a particular instance of an opened device. An example of this is a waveaudio01 device that has a default connection to an ampmix01 device. When the waveaudio01 device is opened, it automatically opens the ampmix01 device, creating an instance of each device. Because devices can be shared in MMPM/2, the waveaudio01 device can be opened again by another application, and two new instances will be created. Although the default connection is the same in both cases, the device-context connections are different.

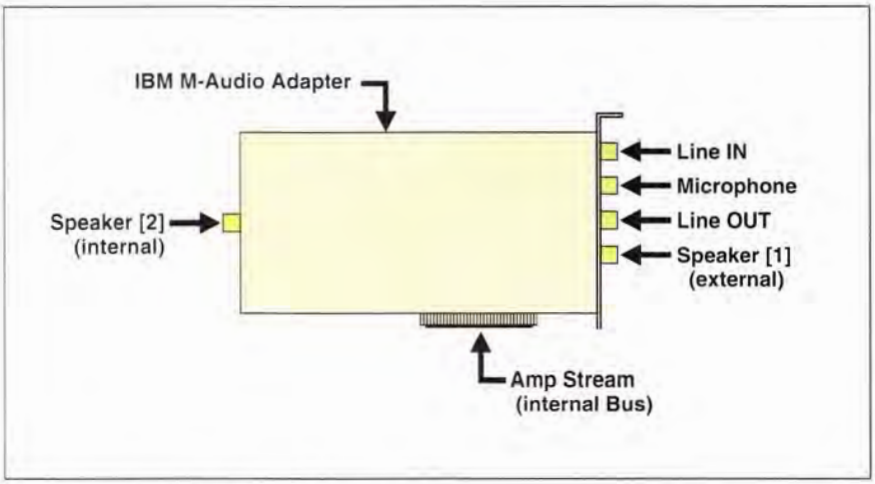



*Figure 1: IBM M-Audio capture and playback adapter*

## AMPLIFIER-MIXER DEVICE

The MMPM/2 amplifier-mixer device is very similar in function to a home stereo amplifier-mixer. Components are plugged into the amplifier-mixer so that audio signals can be transferred to a pair of attached speakers,

headphones, or perhaps another device. The amplifier-mixer serves as the central focus for all audio signals and provides input or output switching as well as the following audio shaping features:

- *Volume.* This feature sets the volume as a percentage of the maximum achievable effect. Volume of the left and right channels for stereo signals can be controlled independently. An OVER parameter can also be specified to cause the volume to fade in or out over a specified period of time.

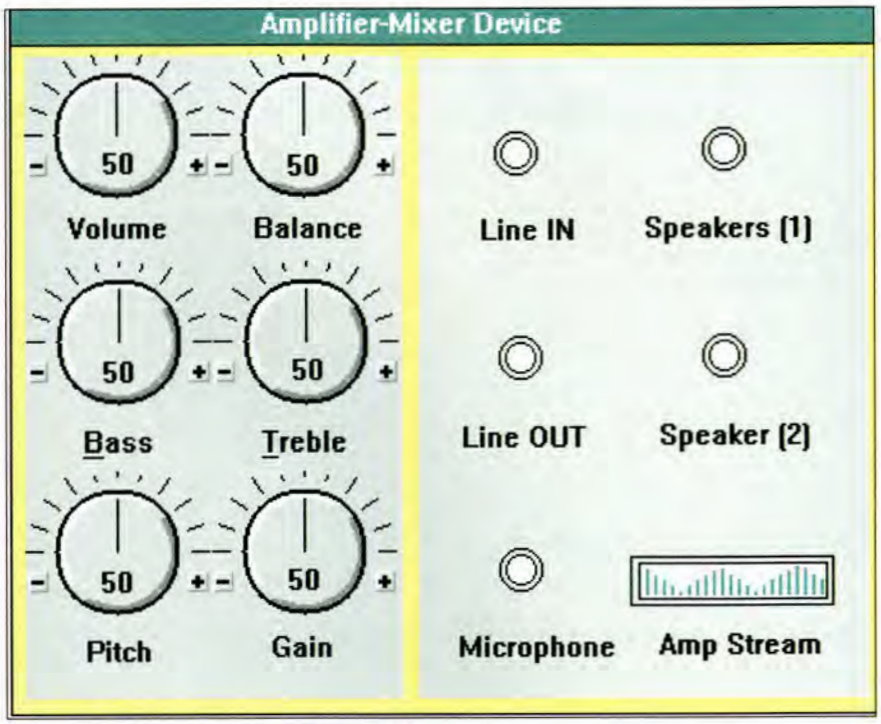



*Figure 2: Amplifier-mixer device software model*

- *Treble.* This feature sets the treble as a percentage of the maximum achievable effect.

- *Bass.* This feature sets the bass as a percentage of the maximum achievable effect.

- *Balance.* This feature sets the final output balance. Zero is defined as full left balance, while 100 is defined as full right balance.

- *Pitch.* This feature sets the pitch as a percentage of the maximum achievable effect.

- *Gain.* This feature sets the gain as a percentage of the maximum achievable effect.

- *Monitor.* This feature controls whether or not the signal from an input device is heard when it is being routed to another device for recording.

Values for all of these functions can be retrieved using the `MCI_STATUS` message, and changed using the `MCI_SET` command. Other companies may develop amplifier-mixer devices for use in MMPM/2 that provide additional capabilities. If an application requests a function not supported, the amplifier-mixer device returns the `MCIERR_UNSUPPORTED_FLAG`.

Figure 2 represents a software model of the function provided by the MMPM/2 logical amplifier-mixer device.

The amplifier-mixer device supports the connection of analog and digital devices. Other MMPM/2 logical devices can be connected to the amplifier-mixer device. For example, the CD audio logical device could provide an analog input to the amplifier-mixer device, which could then be recorded by the digital waveform audio device.

To better understand how the MMPM/2 amplifier-mixer device relates to other logical devices and the underlying hardware, compare the diagram of the M-Audio card in Figure 1 with the amplifier-mixer software model in Figure 2. Although there is actually no visible speaker (2) jack on the back of the audio card, it serves as a convenient fiction for an application to view the PC internal speaker as simply another set of speakers that might be plugged into the back of the audio card. Using the concept of a connector, an application can view all flows of information into and out of the amplifier-mixer device in a similar fashion. Selecting the internal speaker as opposed to the external speakers may require the amplifier-mixer device to issue a completely different set of instructions to the actual hardware device. The application, however, remains completely device independent.

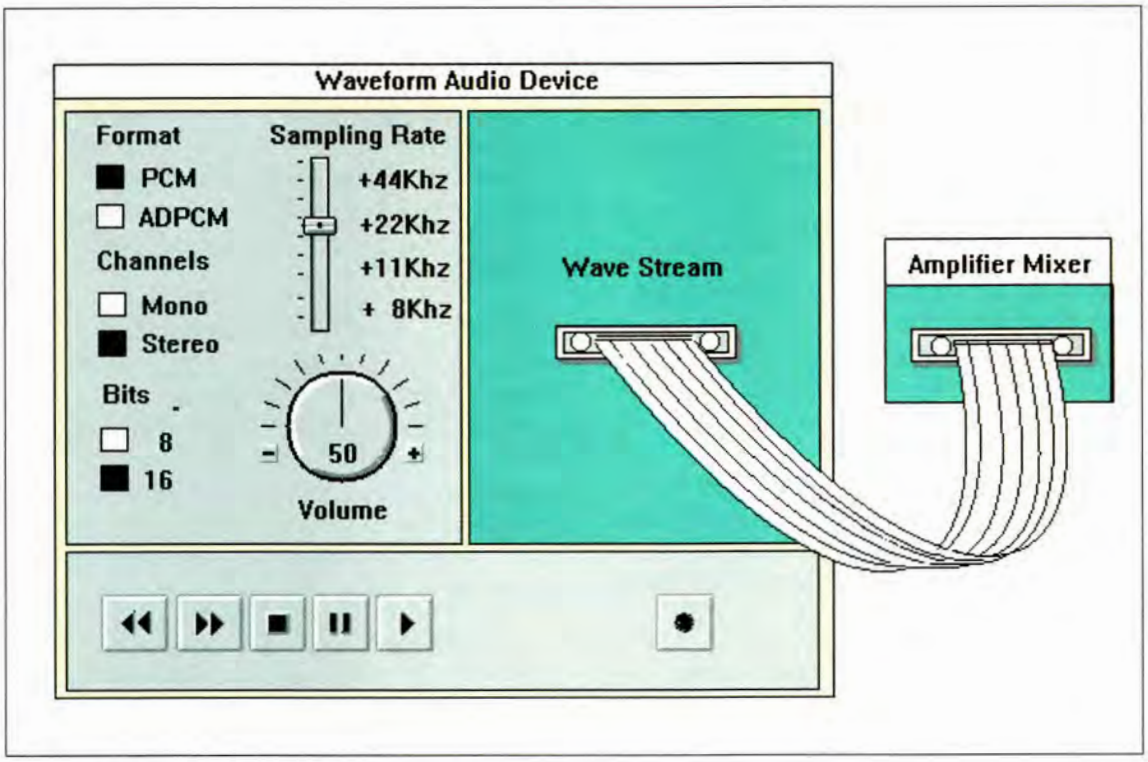



*Figure 3: Waveform audio device software model*

The other features of the amplifier-mixer device may be provided by issuing commands to the hardware device or emulated in software. One example of software emulation is the support of changing the volume over a period of time and is typically referred to as a "fade in" or "fade out." The audio card may only support setting the volume to a particular value, however the amplifier-mixer device can send a series of values over time to achieve the desired effect.

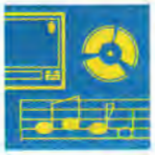
### *The Amp Stream Connector*

The amp stream connector shown in Figure 2 represents the flow of digital information to and from the amplifier-mixer device. Like most home stereo amplifier-mixers, the amplifier-mixer device by itself is not especially notable until another device is attached. At this point, information can be transferred from the device and played back on a pair of attached speakers. The amplifier-mixer device serves as a conduit of information and relies on another device to provide the flow of information. As a result, commands concerned with the transport of information, such as play, seek, or stop, are sent to the attached device, while commands concerned with the transforming of the information, such as treble or bass, are sent directly to the amplifier-mixer device.

As a nicety for applications, the attached device will surface volume control so that the application need not deal with the complexity of an amplifier-mixer device unless some advanced audio functions are required. The volume command will be transparently routed to the attached amplifier-mixer device. If the application needs to talk directly to the amplifier-mixer device, the value of the stream connector can be queried using the `MCI_CONNECTION` message, which returns a device context connection. If the string interface is being used, an alias can be established for the connected device. Amplifier-mixer commands can then be sent directly to the amplifier-mixer device.

Some devices will also provide a "connector service" to eliminate the need to talk directly to the amplifier-mixer device for frequently requested function. An example of this is the waveform audio device, which attempts to process requests for speakers and several other

*The amplifier-mixer device supports the connection of analog and digital devices.*



connector types. If the service is available from the associated amplifier-mixer device it will be routed; otherwise, the function will fail. The connectors and connector services provided by each MMPM/2 logical device are discussed in the *MMPM/2 Programming Guide*.

***Sharing the Amplifier-Mixer Device***

Because many components of MMPM/2 use the amplifier-mixer device, it is typically opened as shareable so that several devices can use the amplifier-mixer device simultaneously or serially in a sharing scheme driven by the application window focus. The MDM is responsible for allocating the resources of the underlying hardware correctly and informing an application with the `MM_MCIPASSDEVICE` message whenever use of the amplifier-mixer device is gained or lost.

*The quality of the waveform can be controlled by setting the format, sampling rate, bits per sample, and number of channels.*

As other media devices request the services of the amplifier-mixer device, the amplifier-mixer can become a source of contention, depending on the capabilities of the underlying audio adapter. For example, the IBM M-Audio adapter can support the simultaneous playback of two mono 22KHz PCM waveforms. However, if a third waveform is started, one of the previous two waveforms must be suspended. The application that opened the waveform audio (waveaudio) device will receive an `MM_MCIPASSDEVICE` message with an event of `MCI_LOSING_USE`. Following completion of the third waveform, the second waveform is automatically restored and can then play to completion.

Luckily, the MMPM/2 system manages all device sharing and only informs the application when the device is temporarily unavailable.

## WAVEFORM AUDIO DEVICE

The MMPM/2 waveform audio device enables an application to play or record digital audio using files or application memory buffers. *Waveform* refers to a digital representation of the original sound wave.

There are two methods used for storing waveform data within a computer system: PCM and ADPCM. Pulse-code modulation (PCM) refers to the variation of a digital signal to represent audio amplitude. This method of assigning binary values to amplitude levels supports the conversion of analog signals to digital signals by adapters such as the IBM M-Audio Capture and Playback Adapter. MMPM/2 recognizes resolutions of the PCM format because it is supported by most audio adapters.

MMPM/2 also recognizes the adaptive differential pulse code modulation (ADPCM) format. ADPCM is a technique for compressing waveform samples. By using the ADPCM rather than PCM format, an application can reduce data storage requirements by a factor of 16 to one; however, some price is paid in fidelity for the higher compression rates.

Figure 3 illustrates the software model for the MMPM/2 logical waveform audio device. The wave stream connector represents the flow of digital information to and from the waveform audio device to its associated amplifier-mixer device. During playback, the waveform audio device sends digitized sounds from either application memory or files to the amplifier-mixer device for subsequent conversion into audio that can be heard through conventional speakers or headphones. When recording, the waveform audio device receives waveforms from the amplifier-mixer device and stores the digital information in a file or in application memory.

Control of the characteristics of the waveform information is provided by the waveform audio device. The quality of the waveform can be controlled by setting the format, sampling rate, bits per sample, and number of channels. The waveform audio device will also let you control the volume. This service is actually provided by the amplifer mixer device in a way that is transparent to the calling application. If other advanced audio shaping features are required, the application can retrieve the device ID of the amplifier-mixer device using the MCI_CONNECTION message. Once the device ID has been obtained, the application can send commands directly to the amplifier-mixer device. Examples include set commands to manipulate treble, bass, and balance.

The following sections describe several typical uses of the waveform audio device. All code examples are provided using the syntax of the MMPM/2 string interface. For more information on the string interface, see the *MMPM/2 Programming Guide* and the *MMPM/2 Programming Reference.*

### *Opening the Waveform Audio Device*

Because the waveform audio device is a compound device, it requires a device element. The device element is typically a file that contains a sampled waveform for playback. The waveform audio device can be opened with or without a device element. A device element can subsequently be specified using the LOAD command, as in Example 1 of Figure 4.

MMPM/2 lets you specify the device to be used for a particular file based on the file's extension or its extended attributes. Using .TYPE extended attributes is the preferred method because they remain with the files even when the files are renamed. Both file extensions and extended attributes can be associated with a device using the multimedia setup application. For instance, assuming files with an extension of .WAV have been associated with the waveform audio device, the command shown in Example 2 of Figure 4 will result in a file being loaded into the waveform audio device.

Finally, the device element and type can be specified to allow an application to specify precisely which device should process the data, as in Example 3 of Figure 4.

### *Recording a Waveform File*

One of the typical uses of the waveform audio device is to digitize an input signal or sound into discrete samples for storage in a file. An example of this would be recording an audio e-mail message. The audio e-mail application would be completely shielded from the complexity of digitizing a signal and would only need to specify a file, while providing the user with a simple control panel to record the message. The user might press a stop button on the control panel when finished describing the idea. The application could then issue a STOP command to the waveform audio device to discontinue the recording, as in Figure 5.

Like many text editors, the waveform audio media driver will not actually modify the original file until it receives a command to save the changes. Any temporary files created during the record operation will be located in the directory specified by the **MSV_WORKPATH** multimedia system variable. The path may be specified on the "system" page of the multimedia setup application. The use of temporary files is completely transparent to the application.

**Example 1: Specifying only the device name**

```
open waveaudio alias wave shareable
load wave c:\mysounds\train.wav
```

**Example 2: Specifying only the device element**

```
open c:\mysounds\monkey.wav alias monkey shareable
```

**Example 3: Opening a specific device**

```
open c:\mysounds\bells.wav type waveaudio alias wave shareable
```

**Example 4: Specifying the readonly option**

```
open bigwave.wav type waveaudio alias wave readonly shareable
```

*Figure 4: Opening the waveform audio device*

```
open myidea.wav waveaudio alias wave wait
record wave notify
stop wave wait
save wave wait
close wave wait
```

*Figure 5: Recording a waveform file*

The file can be saved using the original file name or a new file name with the **SAVE** command. If a **SAVE** command is not issued before closing the waveform audio device, all changes will be discarded:

It is possible to open or load the waveform audio device specifying the readonly option. In this mode, the waveform audio device

prevents any modification to the file from **SAVE** or **RECORD** commands. In certain circumstances, the driver can optimize performance if it knows that the file will not be modified. The readonly option also lets multiple applications share the same file for playback purposes and prevents inadvertent modification of the file, as shown in Example 4 of Figure 4.

### *Creating New Files*

The waveform audio device will create a new file with the **OPEN** or **LOAD** command when a file name is specified that does not exist. To support file creation from the string interface, a special file name called NEW is reserved for system use. This file name should be used in place of the usual application-supplied file

```
open new type waveaudio alias wave wait
record wave notify
stop wave wait
save wave myspeech.wav wait
close wave wait
```

*Figure 6: Creating a file from the string interface*

```
set wave format tag pcm wait
set wave samplespersec 22050 wait
set wave bitspersample 8 wait
set wave channels 1 wait
```

*Figure 7: Specifying the waveform format*

name. The **SAVE** command must be issued to give the file a permanent name, as in Figure 6.

When a file is initially created, default settings are assigned by the media driver and may depend on the capabilities of the audio adapter. The IBM waveform audio driver will use 16-bit mono, 22KHz PCM as the default for 16-bit adapters. If the adapter does not support 16-bit PCM, then the resolution will be downgraded to eight bits.

An application should always set the sampling rate, resolution, waveform format, and number of channels to ensure that the waveform is created with the desired parameters, as in Figure 7.

When modifying the settings on a waveform audio device, the format should be changed first because it may force the automatic modification of other settings to make them compatible with the new format. For instance, a waveform audio device that supports 16-bit PCM may only support 8-bit ADPCM. Changing the format from PCM to ADPCM will automatically change the bits-per-sample setting.

### *Playing and Recording non-RIFF Waveforms*

The waveform audio device will create new waveforms according to the resource interchange file format (RIFF) WAVE data standard. It is possible, however, to play other data formats if a compatible I/O procedure has been supplied. Selection of the appropriate I/O procedure is transparent to the application if it is installed. If you are interested in writing a compatible I/O procedure, see the *MMPM/2 Programming Guide*.

One example of this feature is MMPM/2's ability to play waveform audio files that were created using IBM's Audio Visual Connection™ (AVC) application and the M-Audio card. The AVC support provides playback capabilities only. The waveform audio device will temporarily report **FALSE** to the save and record capabilities of the device capabilities (**MCI_GETDEVCAPS**) function when the underlying I/O procedure does not support the creation of files. Applications should check the device capabilities to appropriately display a user interface that reflects the true capabilities of the waveform audio device and its associated I/O procedure.

For example, a waveform editor application should grey out its record button when an AVC file is loaded, as only playback operations are supported. Querying the device capabilities would return **FALSE** for "can record." If a waveform file in the RIFF WAVE format is subsequently loaded, the record button should be enabled, because the "can record" query now returns **TRUE**. In all instances, by using the high-level MMPM/2 **mciSendString** or **mciSendCommand** interface to reference device capabilities, the application is shielded from the underlying implementation.

### *Using Memory Buffers*

Specialized applications such as a waveform

editor may require the capability of playing and recording using application memory buffers instead of files. The memory playlist feature of MMPM/2 provides the construct for supplying memory buffers to the waveform audio device. Besides implementing simple circular buffering schemes, memory playlists can be used to synthesize complex and unique waveform sounds. The chime application in the *MMPM/2 Sample Programs Workbook* shows an example of a waveform that is actually constructed of several different waveforms. By following each **DATA** statement with a **MESSAGE** statement, an application can be informed as to when the buffer can be reused.

Depending on the complexity of the application, memory playlists can be used to provide a single large memory buffer or multiple buffers in a circular buffering scheme. Following is an example of how a memory playlist might be constructed to implement a simple circular buffering scheme:

```
0: NOP
1: DATA...
2: MESSAGE...
3: DATA...
4: MESSAGE...
5: DATA...
6: MESSAGE...
7: BRANCH 0
```

Whether the playlist is being used for play or record operations, the **MESSAGE** instruction will notify the application when the playlist processor has consumed or filled the preceding **DATA** buffer. An **MM_MCIPLAYLISTMESSAGE** will be sent to the window procedure specified when the waveform audio device was originally opened.

### *Waveform Audio Commands*

Following are descriptions of the String Interface commands used by applications to control the recording and playback of waveform data. Analog input devices for recording waveforms are a microphone and tape deck. Analog output devices for waveform playback are a stereo amplifier or speakers connected to an audio adapter.

- **OPEN** initializes the waveform audio player.
- **CAPABILITY** gets device capabilities.
- **CUE** cues the device for minimum delay in recording or playback, for example, cuing a wave input or output device.
- **RECORD** records waveform data, for example, the position to record to and from.
- **PLAY** plays back waveform data by means of an audio adapter, for example, the position to play to and from.
- **PAUSE** suspends the current play or record action.
- **RESUME** resumes the current play or record action from a paused state.
- **SEEK** seeks to a specified location.
- **SET** sets device information for recording, playback, and saving, for example, changing time formats to samples, time formats to bytes, format tags, number of channels, number of bits per sample, average number of bytes per second, number of samples per second, and block alignment.
- **STATUS** gets status on device information, for example, for format tags, channel counts, block alignment, number of samples per second, number of bits per sample, average number of bytes per second, and record or playback level.
- **INFO** gets the name of the currently loaded file.
- **STOP** stops the waveform device before loading a new file.
- **LOAD** loads a waveform data file.
- **SAVE** saves the device element in its current format.
- **SETCUEPOINT** sets a cuepoint.
- **SETPOSITIONADVISE** sets a position change notification request.
- **SETSYNCOFFSET** sets a synchronization offset.

*Depending on the complexity of the application, memory playlists can be used to provide a single large memory buffer or multiple buffers in a circular buffering scheme.*

- `CLOSE` closes the waveform audio player.
- `CONNECTOR` enables or disables a connector, queries its state, or identifies its type.

**John E. Parsons,** *IBM Corp., Personal Systems Programming Division, P.O. Box 1328, Internal Zip 1508, Boca Raton, Fla. 33429. Parsons is an advisory programmer in the Multimedia Software Design department and was the lead designer for the initial release of MMPM/2. He has been with IBM for two years and is currently the project lead for the video extensions to MMPM/2. Parsons received a B.S. and Masters in computer science from the Georgia Institute of Technology where he is also a qualified Ph.D. candidate. Previous to IBM, Parsons worked for Harris Corp. as a lead software engineer in the development of advanced communications and teleconferencing systems.*

**Val Enright,** *IBM Corp., Personal Systems Programming Division, P.O. Box 1328, Internal Zip 1606, Boca Raton, FL 33429. Enright is a staff information developer in Information Development and has been with IBM for 12 years. Her responsibilities include technical writing of OS/2 programming information. Her current assignment is the video extensions to MMPM/2.*

## REFERENCES

Multimedia Presentation Manager Toolkit/2 publications can be ordered by calling 1-800-IBM-PCTB. Following are the toolkit publication prices and order numbers:

*MMPM/2 Programming Guide* —Describes application and subsystem programming interfaces to help you select and implement functions for OS/2 multimedia applications. $23.50 (IBM Doc. S41G-2919).

*MMPM/2 Programming Reference* —Provides detailed information on multimedia functions, messages, and data structures to enable you to write code for multimedia applications. $22.00 (IBM Doc. S41G-2920).

*MMPM/2 Sample Programs Workbook* — Gives code examples from MMPM/2 sample application programs and templates for subsystem components. $20.50 (IBM Doc. S41G-2921).

*CUA Guide to Multimedia User Interface Design*—Describes design concepts to consider when designing a CUA multimedia interface. $9.70 (IBM Doc. S41G-2922).

*OS/2 Multimedia Advantages* —Describes advantages offered by OS/2 and MMPM/2 multimedia platform. $3.70 (IBM Doc. S41G-2923).

Complete MMPM/2 Technical Library — $78.75 (IBM Doc. S41G-3321).

Multimedia

# IBM ActionMedia II Support For MMPM/2

*by Mary Alvarez and Val Enright*

*Application developers who use the Multimedia Presentation Manager/2 (MMPM/2) programming toolkit are already acquainted with the multimedia device-independent support made available to OS/2 applications with MMPM/2 1.0. The 1.0 support, primarily for audio devices such as the IBM M-Audio Capture and Playback Adapter, is described in the article "Multimedia Waveform Audio Support," pages 101-109 of this issue.*

*With the implementation of the MMPM/2 Media Control Interface for IBM ActionMedia II multimedia adapters, digital video is added to the list of devices supported for MMPM/2 applications. This support is available as a package that includes the ActionMedia II adapters and the Media Control Interface driver, which is installed with the MMPM/2 installation program.*

## ACTIONMEDIA II DIGITAL VIDEO DEVICE FOR MMPM/2

The ActionMedia II digital video device for MMPM/2 supports playback and recording of digital video interactive (DVI) audio-video subsystem (AVSS) format motion video and capture and display of AVSS-format images. Figure 1 provides a software model of media control interface functions supported by the ActionMedia II digital video device.

An AVSS-format file contains one or more streams of data. Each stream contains digital data to describe the playback of a single audio or video stream. There can be several such streams, all intended for simultaneous playback. To reduce head movement on the storage device, data from the various streams is interleaved in frames.

The ActionMedia II digital video device supports data streaming of motion video from AVSS-format files on any device that can appear as a logical drive, such as a hard disk, CD-ROM drive, or network file server. Streaming of video data to and from system memory is not currently supported.

## DIGITAL VIDEO DEVICE CONTEXTS

The ActionMedia II device supports multiple instances of the digital video device, known as device contexts. Multiple video and still image device contexts can be displayed simultaneously. Only one video device context can be playing, recording, or monitoring at a time. If more than one video device context is displayed, the one in the window with the input focus is active and the others are paused.

Multiple still image device elements can be open simultaneously, and stills can be open while a video instance plays a motion video file or monitors live video. However, no other still or video device contexts can be open while a recording instance is open.

Taking these restrictions into consideration, Figure 2 shows the combinations of loaded video and image elements allowed by the ActionMedia II digital video device at any given time. The numbers and combinations shown in the table are based on the 2MB of video RAM (VRAM) on the ActionMedia II display adapter.

## DIGITAL VIDEO PM WINDOWS

The ActionMedia II device displays digital video and image device contexts in PM

*Val Enright*

windows. As shown in Figure 2, up to five digital video windows can be displayed simultaneously. The ActionMedia II digital video device supports two types of window, default and application-defined. The default window is used to display motion video and

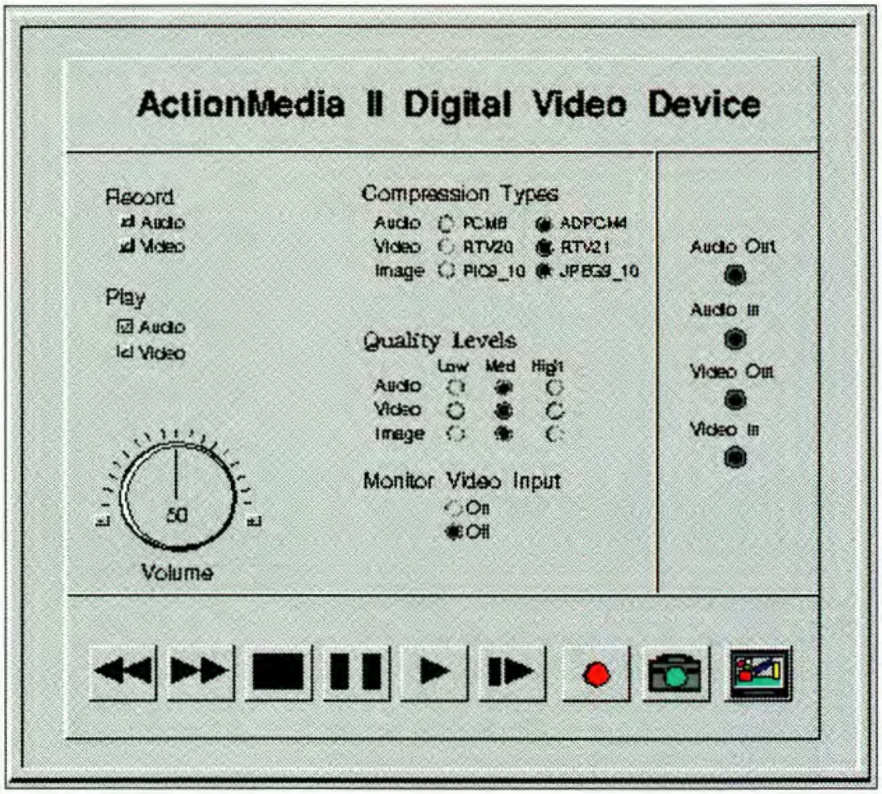



*Figure 1: Software model of the ActionMedia II digital video device*

| VIDEO ELEMENT | IMAGE ELEMENT |
|---|---|
| 0 | 5 |
| 1 | 3 |
| 2 | 2 |
| 3 | 1 |
| 4 | 0 |

*Figure 2: Video and image element combinations*

images if no other window handle is specified. It is a standard desktop window that can be moved, sized, and minimized by the user. When a window is minimized, the icon is a scaled-down version of the video that continues to be updated at the rate of 30 frames per second.

Assuming the source DVI video is 256x240 pixels, the default window size is 320x240 VGA pixels on a VGA display, and 640x480 XGA pixels on an XGA display. The position of the default window is determined by the Presentation Manager shell.

The default window has a default size option on the pull-down system menu. This option resizes the window so the image or video in the client area is displayed at its default size in a high-resolution view.

An application-defined window can be used when an application requires more control over a window. The application has the freedom to place video in its own client area or in a child window, to add menus, and so on. If the parent of the application-defined window is not the PM desktop, its parent window handle must be supplied with the open request.

If an application specifies a parent window handle when opening a device, it must close the logical video device before destroying the parent window. Otherwise, video can continue to be keyed into the area where a child window was located.

When an application-defined window is used to display video or still images, the driver subclasses the window to ensure correct video updating. Subclassing the window:

- Positions and sizes the window so that its boundaries are coincident with DVI pixel boundaries
- Ensures that the window cannot be larger than the maximum size of the displayed video or image
- Modifies the tracking rectangle so the user cannot drag the size border beyond the maximum size of the displayed video image.

Whether the default window or an application-defined window is used for output, the digital video device confines its output to the device coordinates of the specified window. In addition, the device restricts its output (and overwriting) to pixels set to the logical transparency color within the coordinates. This lets the PM application

continue to draw graphics or output text into a window so the output overlays a video image.

## PLAYING MOTION VIDEO FILES

By default, playback of digital motion video is displayed in a window supplied by the ActionMedia II digital video device. The default window is created when the device is opened but is not made visible until a `CUE`, `PLAY`, `SET MONITOR ON`, `RECORD`, or `RESTORE` command is issued. Example 1 in Figure 3 illustrates playing an entire motion video AVSS format file. Because `TO` and `FROM` parameters are not specified with the `PLAY` command, the file is played from the current position to the end of the file. When a motion video device element is opened, the current position in the media is the first playable area after any header or table of contents information.

Each frame in a motion video file is associated with a number. From the perspective of the digital video device, each file is zero-based. The first frame is frame 0, the second frame is frame 1, and so on. The number of the last frame in a file is one fewer than the total number of frames in the file. The current position always indicates the frame about to be displayed, rather than the frame that is currently displayed.

When a play position in motion video is specified with the `FROM` parameter of the `PLAY` command, the actual position reached is accurate only to the nearest intracoded frame (I-frame). A position specified with the `TO` parameter, however, is exact. To specify an exact position in the video file to begin play, use the `SEEK` command, which moves the current position in a file to an exact point. Example 2 in Figure 3 illustrates moving the current position to frame 100 and then playing to frame 2000.

## RECORDING MOTION VIDEO

The ActionMedia II digital video device supports recording of motion video into new and existing AVSS-format video files. Supported compression algorithms for recording video are real time video (RTV) 2.0 and 2.1, with RTV 2.1 the default setting for video compression. An associated sound track can be recorded in the pulse code modulation 8-bit format (PCM8) or the adaptive differential pulse code modulation 4-bit format (ADPCM4). ADPCM4 is the default setting for audio compression.

Video can be input at either American or European television broadcast standards. The American standard is the National Television

**Example 1: Playing a motion video file**

```
open seaworld.avs type digitalvideo alias myvideo wait
play myvideo wait
close myvideo
```

**Example 2: Playing a portion of a motion video file**

```
open seaworld.avs type digitalvideo alias myvideo
set myvideo time format frames wait
seek myvideo to 100 wait
play myvideo to 2000 wait
close myvideo
```

**Example 3: Recording video into an existing file**

```
open digitalvideo alias myvideo wait
set myvideo video quality low wait
set myvideo audio quality low wait
set myvideo time format frames wait
set myvideo monitor on wait
record myvideo to 99 wait
play myvideo wait
save myvideo oldvid.avs video wait
close myvideo
```

**Example 4: Recording only audio into a new AVSS-format file**

```
open digitalvideo alias myaudio wait
set myaudio video record off wait
set myaudio audio compression pcm8 wait
set myaudio audio quality high wait
set myaudio time format hms wait
cue myaudio input wait
record myaudio to 00:10:00 wait
play myaudio wait
save myaudio newaud.avs wait
close myaudio
```

*Figure 3: Playing and recording motion video*

Standard Committee (NTSC), and the European standard is Phase Alternation by Line (PAL).

Recording is supported for new and existing files. Example 3 in Figure 3 illustrates recording live video. Monitoring is set on so the incoming video signal can be viewed in the default video window before it is recorded. Live video can also be monitored without recording. During recording, digital audio and video data is stored in a temporary file created for the video device element. In Example 3, the device element is played back after recording is finished so the video can be viewed before it is saved as a permanent file. To save the data in an AVSS-format video file, specify the name of a new or existing file with the **SAVE** command. In this example, the device element is saved to an existing file. The save operation completely overwrites any data in the existing file with newly recorded data from the device element.

**Example 1: Displaying an image file**

```
open manatee.avs type digitalvideo alias myimage wait
restore myimage wait
close myimage
```

**Example 2: Displaying Multiple Image Files**

```
open manatee.avs type digitalvideo alias image1 shareable wait
open dolphin.avs type digitalvideo alias image2 shareable wait
open whale.avs type digitalvideo alias image3 shareable wait
open snapper.avs type digitalvideo alias image4 shareable wait
open catfish.avs type digitalvideo alias image5 shareable wait
restore image1 wait
restore image2 wait
restore image3 wait
restore image4 wait
restore image5 wait
close image1
close image2
close image3
close image4
close image5
```

**Example 3: Capturing a Still Image from Live Video**

```
open digitalvideo alias vid shareable
set vid monitor on wait
capture vid wait
restore vid wait
save vid birds.avs image wait
close vid
```

**Example 4: Capturing a Still Image from a Video File**

```
open motion.avs type digitalvideo alias myvideo wait
set myvideo image compression jpeg9_10 wait
play myvideo notify
capture myvideo wait
restore myvideo wait
rewind myvideo wait
play myvideo notify
capture myvideo wait
restore myvideo wait
save myvideo image mystill.avs wait
close myvideo
```

*Figure 4: Displaying and capturing still images*

The ActionMedia II digital video device supports high, medium, and low quality settings for video, audio, and still images. With these settings, an application can alter the supported compression algorithms. In Example 3, low levels are specified for audio and video; audio and video quality are sacrificed for low data rates. The default setting for audio, video, and image is medium quality, or the nominal CD-ROM rates.

Applications can specify that only video or only audio is to be recorded. In Example 4 of Figure 3, video recording is turned off so only audio will be recorded. The audio quality for recording PCM 8-bit digital audio is set to high. The best quality audio will be produced at the expense of a high data rate. Specifying the **CUE** command readies the device for recording and makes the default window visible. The audio data is saved into an AVSS-format video file.

## DISPLAYING IMAGE FILES

The ActionMedia II digital video device supports the display of DVI AVSS-format still image files. PIC 1.0 9-bit and 16-bit compression formats are supported for display, as is the Joint Photographic Experts Group (JPEG) 1.0 9-bit format. With the exception of PIC 1.0 16-bit, these formats are also supported for image capture.

In Example 1 of Figure 4, issuing the **RESTORE** command transfers the **MANATEE.AVS** data object

from the device element temporary buffer to the display surface. In the example the device element is loaded from an image file specified with the OPEN command.

Because ActionMedia II supports multiple device contexts, up to five still image device elements can be open at one time, as shown in Example 2 of Figure 4.

## CAPTURING A STILL IMAGE

The ActionMedia II driver captures DVI AVSS-format still files from live video and from replayed digital video files. To capture a still, monitoring of an incoming video signal must be set on or a DVI AVSS-format file must be in playback.

Example 3 in Figure 4 shows how to capture a still image from live video. Monitoring is set on so the live video can be viewed in the default window. The CAPTURE command stores the current image as a device element in a temporary file. The RESTORE command lets you view the device element before it is saved as a permanent image file with the SAVE command. To save a device element as an image file, specify the IMAGE parameter.

The temporary capture file is located in the subdirectory specified by the multimedia work path. This path can be set with the Multimedia Setup program.

Example 4 of Figure 4 demonstrates capturing a still from a motion video file:

- The first still is captured as a device element and displayed.
- The motion video file is repositioned to the beginning and played again so another capture can be made.
- The second capture overwrites the first capture in the temporary buffer provided for the device element.
- The device element is saved in a file.

In Example 4 the image compression format is set to the JPEG 1.0 9-bit compression format.

The default setting for image compression is the PIC 1.0 9-bit format.

## DIGITAL VIDEO COMMANDS

The following Media Control Interface commands are supported by ActionMedia II:

- CAPABILITY requests information about the digital video device.
- CAPTURE captures the current video image.
- CLOSE closes the device element and associated resources.
- CONNECTOR queries the status of digital video device connectors.
- CUE prepares the device for playback or recording.
- INFO returns names of files loaded for the current device context.
- LOAD loads an AVSS-format file for the current device context.
- OPEN creates a device context.
- PLAY signals the device to begin playing.
- RECORD signals the device to begin recording.
- RESTORE transfers an image device element to the display surface.
- RESUME resumes a recording or playback operation.
- REWIND changes media position to the first playable position.
- SAVE saves the currently loaded device element in a file on disk.
- SEEK changes the position in the media.
- SET sets digital video device information.
- STATUS queries digital video device information.
- STEP steps the digital video device forward one frame.

*Because ActionMedia II supports multiple device contexts, up to five still image device elements can be open at one time.*

**Mary Alvarez,** *IBM Corp., Entry Systems Technology, P.O. Box 1328, Internal Zip 1218, Boca Raton, Fla. 33429. Alvarez is an advisory programmer in the advanced digital video development department at IBM. She led the release of the ActionMedia II Media Control Interface for MMPM/2. She has been with IBM for 11 years and is currently the project lead for digital video software development. Alvarez received a B.S. in mathematics education and an M.S. in management science, both from the University of Miami.*

**Val Enright.** *IBM Corp., Personal Systems Programming, P.O. Box 1328, Internal Zip 1606, Boca Raton, Fla. 33429. Enright is a staff information developer in Information Development and has been with IBM for 12 years. Her responsibilities include technical writing of OS/2 programming information. Her current assignment is the video extensions to MMPM/2.*

## REFERENCES

*IBM International Technical Support Centers: IBM Personal System/2 Multimedia Fundamentals* (IBM Doc. GG24-3653)

*IBM ActionMedia II Technical Reference* (IBM Doc. 04G5144)

*IBM ActionMedia II Programming Guide and Reference for Multimedia Presentation Manager/2* (IBM Doc. 42G2048)

## ACTIONMEDIA II UPGRADE KITS

Customers who purchased either the ActionMedia II Display Adapter/A 2MB or ActionMedia II Display Adapter 2MB before September 30, 1992 and want to upgrade their systems with the ActionMedia II digital video support provided for OS/2 2.0 with MMPM/2 can order an upgrade kit that includes:

- Audio Video Kernel (AVK) Version 1.1 device drivers
- Media Control Interface libraries for OS/2 2.0

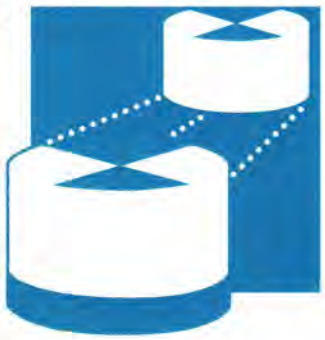
To receive the upgrade, you can mail the front page of the AVK device driver documentation or toolkit proof of license to:

IBM Fullfillment Center
P.O. Box 4263
Crawfordville, IN 47933

Or you can fax the device driver page or toolkit proof of license to Fred Fuhrmann at the Fullfillment Center at (317) 364-0787. Be sure to include your name and shipping address.

For more information, call the IBM Fulfillment Center at 1 (800) 845-4263.

Database Manager



# Driving Forward with the Database Manager Command Line Interface

*by Theodore Shrader and Joel Dunn*

*OS/2 Extended Services (ES) offers a new and easy-to-use interface to Database Manager functions with the DBM Command Line Interface. Database commands can now be issued directly from a windowed or full-screen OS/2 prompt or as part of a command file. The syntax of each command closely follows database APIs, and on-line help is available for questions on syntax, semantics, and messages. This article describes what the DBM Command Line Interface is, how it works, and how developers and administrators can use the interface on their workstations. It also provides an example that centers around the use of this interface with forward recovery, a new feature in this release of the Database Manager component of Extended Services.*

## TYPES OF INTERFACES

To access Database Manager functions, three types of interfaces are available. Application programming interfaces (APIs) reside at the lowest level. Developers can write applications in C, COBOL, FORTRAN, and REXX using these APIs to access database functions. Applications can easily invoke static or dynamic SQL statements after running their application code through precompilers supplied with the Database Manager. Although APIs offer the most options, they are not feasible for all users, such as those without programming experience or those who cannot devote the time necessary to create the applications.

For those who do not wish to write programs calling APIs, graphical user interfaces offer a simplified yet powerful form of access to the Database Manager. Query Manager™ and the Database Administrator (DBA) Tools™ user interfaces are included with Extended Services. They provide query and report functions as well as administrative capabilities, without users needing to know the details of the underlying APIs. This functionality allows developers and users to devote their resources to other tasks.

Lastly, command line interfaces provide a flexible middle ground between APIs and graphical user interfaces. The DBM Command Line Interface can be executed after any OS/2 prompt. Its commands can be included in command files, and it provides an easy way to access database functions without users needing to write sophisticated programs or having to deal with the overhead of graphical front-ends.

## DBM COMMAND-LINE SYNTAX AND SEMANTICS

Each command is preceded by the `DBM` prefix followed by command options (if any) and ending with the command statement to execute. For those who are familiar with the Database Manager APIs, the command statements have variable fields that correspond to those required by the API function calls. The DBM Command Line Interface brings almost all the flexibility and power provided by those APIs to the OS/2 prompt. In addition to the APIs, SQL data definition language (DDL) and data manipulation language (DML) statements can be issued to create or drop table and view objects; grant or revoke authorizations; insert, update, delete, and select rows, as well as to perform other actions.

*Theodore Shrader*

*Joel Dunn*

## COMMAND OPTIONS

### *-R and -O (Report File Generation and Output Inhibit)*

DBM commands can be invoked from the OS/2 prompt, and the output of the command can be redirected to a file using the operating system technique of file redirection (`> filename`) or with the DBM command options: `-R` for a report file and `-O` for output inhibit:

```
DBM SELECT * FROM ORG > C:\ORG.RPT
DBM -R(C:\SQLLIB\ORG.RPT) SELECT * FROM ORG
```

```
1992-01-27-15.04.14

  Query Report for:  SELECT * FROM ORG

DEPTNUMB  DEPTNAME        MANAGER  DIVISION    LOCATION
--------  --------------  -------  ----------  -------------

      10  Head Office         160  Corporate   New York
      15  New England          50  Eastern     Boston
      20  Mid Atlantic         10  Eastern     Washington
      38  South Atlantic       30  Eastern     Atlanta
      42  Great Lakes         100  Midwest     Chicago
      51  Plains              140  Midwest     Dallas
      66  Pacific             270  Western     San Francisco
      84  Mountain            290  Western     Denver

  8 record(s) selected.
```

*Figure 1: Example query report*

```
1992-01-27-14.27.44   DBM start using database sample
1992-01-27-14.27.47   START USING DATABASE completed
                       successfully.

1992-01-27-14.57.20   DBM select count(*) from staff
1992-01-27-14.58.42   SELECT completed successfully.

1992-01-27-15.04.14   DBM -r select * from org
1992-01-27-15.04.28   SELECT completed successfully.

1992-01-27-16.27.25   DBM stop using database
1992-01-27-16.27.40   STOP USING DATABASE completed
                       successfully.
```

*Figure 2: Example history file*

Both of these commands send the report generated by the `select` statement to a file called ORG.RPT. However, while the first command directs all output to a file, the `-R` option in the second command sends the data to the file and screen (`stdout`). (It acts much the same way as the `tee` command in UNIX.) Report files generated by the `-R` option will also have a header in the report file. This header consists of the timestamp and the query executed as a reminder of when and which `select` statement created the report. Figure 1 shows an example query report. (Used in combination, the `-R` and `-O` options can perform the same action as the operating system technique of file redirection, but with the report header.)

If a file name is not specified with the `-R` option, the default report file, DBM.RPT, will be created in the current directory. If the report file already exists, the new data will be appended to the file. If a file name is specified, it must be contained within parentheses. If the file name does not contain a path, the current path will be used. The `-R` option is only valid on `SELECT`, `GET`, `LIST`, `REORGCHK`, and `ROLLFORWARD` commands. A warning message will be generated if `-R` is used with other commands, but the commands will still be executed.

### *-C And DBMCOMMIT (Auto-Commit)*

After each command is executed, the DBM Command Line Interface will issue a commit statement. This feature may not be desirable for users who want a series of commands to be considered as a single unit of work. To get a series of commands to either all succeed or all fail, the `-C` option should be used on every command in the series. This option allows users to inhibit the auto-commit feature:

```
DBM -C INSERT INTO ORG (xxx) VALUES (xxx)
```

For users who have a large number of commands in a unit of work, the `DBMCOMMIT` environment variable could be used:

```
SET DBMCOMMIT=NO
```

This variable may be used for a single session at the OS/2 command prompt or for all sessions by placing this statement in the CONFIG.SYS file. A value of `NO` will inhibit the auto-commit done by the DBM Command Line Interface. However, with this variable set to `NO`, users must remember to issue a `COMMIT` or `ROLLBACK` themselves. The DBM will automatically `COMMIT` any uncommitted data when a `STOP USING DATABASE` command is issued.

### *DBMHIST (History file creation)*

Every time the DBM Command Line Interface executes a command, it is logged in the DBM.RUN file. This file is located in the \SQLLIB directory where the DBM resides. This history file will not exceed 32K. When this boundary is reached, it will erase the file, start over, and continue logging commands. An example of the information placed in the history file is shown in Figure 2. Just as the auto-commit feature can be disabled, the history file can be disabled by setting the environment variable `DBMHIST` to `NO`.

### *\ (Continuation of long input statements)*

Statements entered from the command line are limited by OS/2 to 254 characters. Entering long statements can be a problem when users have complicated `select` statements to issue. There are some environments where this limitation can be overcome. In OS/2 command files, a statement cannot span multiple lines. In REXX command files, a comma at the end of a line signals that the statement will continue on the next line. If a user enters a DBM command at an OS/2 prompt and it is too long for the line, the user can leave a space followed by a backslash at the end of the command before pressing the Enter key. A DBM command prompt will appear, and the user can continue typing the command. Figure 3 shows an example of the continuation character with message help.

### *-H and ? (Help)*

If users need help, typing in `DBM ?` will provide general help and a list of the commands supported by the DBM Command Line Interface. If `DBM ?` is followed by a keyword or phrase, it will bring up on-line help for the appropriate command. If a message number follows the question mark, an on-line explanation of the message will appear. Users can also browse through the help document to view the commands available along with their syntax and description. For example, the following statement calls the on-line help text for the `CATALOG DATABASE` command:

```
DBM ? CATALOG DATABASE
```

If a SQL or DBM warning or error message is returned on a command, such as message number SQL1024, more help can be obtained by typing:

```
DBM ? SQL1024
```

The on-line help for message `SQL1024` will be presented. Help is available for all messages generated by DBM applications (including Query Manager and the DBA Tools).

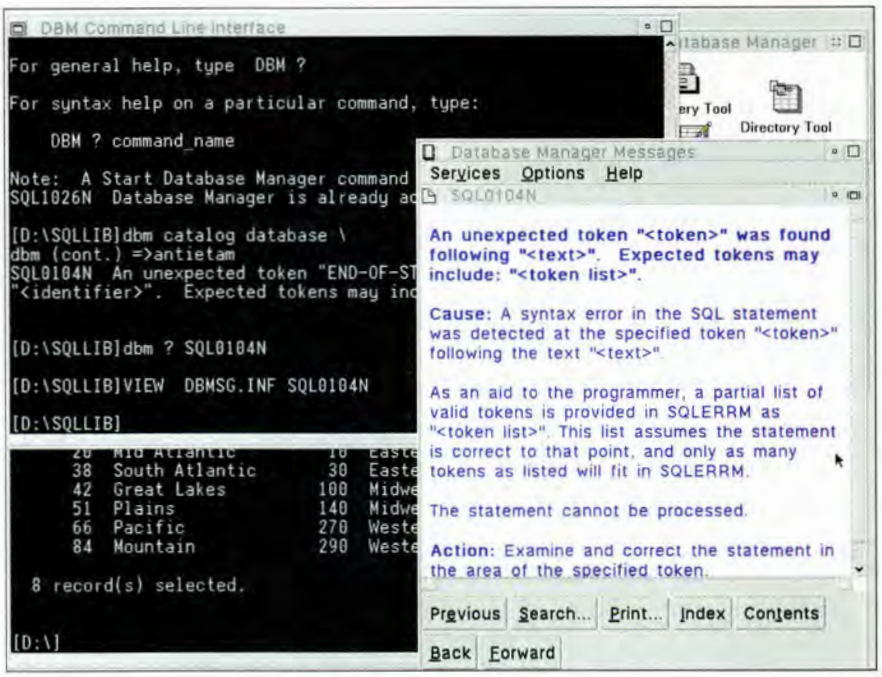



*Figure 3: DBM error with message help in background*

## WRITING A COMMAND FILE

The DBM Command Line Interface is written almost entirely in REXX, and as a result, it complements the command files developers and users have created. Users can write simple command files or more complex ones that loop and execute code based on various conditions. This section presents an example REXX command file using the DBM Command Line Interface.

At some time, database users will need to extract data from a table, but they may not always remember the exact statement. With the DBM Command Line Interface, users can create command files for commands that need to be executed repeatedly. For example, a user or developer may connect to a database and not remember the name of the table containing the desired data or remember the columns that need to be part of the `select` statement. The REXX command file in Figure 4 can accomplish this task.

```
/*  Example using the Command Line Interface:   LISTMY.CMD */
parse upper arg nameof keyword namein trash
trace off

select

  when nameof = 'TABLE' | nameof = 'TABLES' then
  do
    select

      when keyword = '' then
      do
        say ' '
        say 'REPORT for:  LISTMY 'nameof  keyword  namein
        say ' '
        call dbm "select creator, name, type, remarks from sysibm.systables",
                 "where creator not like 'SYSIBM%'"
      end
```

*Figure 4: Source listing for LISTMY.CMD* *(continued on page 121)*

*Instead of recovering to the last commit point, users can recover to a specific point in time.*

This command file accepts three input values; the arguments are folded to upper case. NAMEOF refers to the type of object the user wishes to query: a TABLE, COLUMN, or INDEX. The contents of NAMEOF determines where the code will flow, and, if its value is not recognized, general help will be given. KEYWORD is optional, and in this example it can be ALL or ON. NAMEIN is the name of the table whose columns or indices a user wishes to query. TRASH is just a place holder for extraneous or unexpected information. If TRASH has a value, it will be ignored without a warning.

## ROLL-FORWARD RECOVERY EXAMPLE

Using the DBM Command Line Interface, developers and system administrators can easily access the DBM's ability to roll data forward to a point in time or to the end of the database's recovery logs. Rolling data forward allows developers to quickly recover their work after data has been inadvertently lost. This situation differs from a crash where a user has connected to a database and added data, and the workstation on which the database resides loses power. In crash situations, the database must be restarted, such as with the following command:

```
DBM RESTART DATABASE dbname
```

This action recovers the database to the last committed checkpoint. Any uncommitted changes are rolled back and removed from the database.

Roll-forward recovery allows users to specify a finer granularity of where the database recovers. Instead of recovering to the last commit point, users can recover to a specific point in time, such as a previous day's work, or to the end of the transaction logs.

This feature will be particularly useful to application developers and system administrators. The following example illustrates such a scenario. A developer created a database to hold sampling data for use with an application. This data could be inputs gathered from a sensor registering readings at periodic intervals. Unless the database was enabled for roll-forward recovery, the recorded sensor data would be lost when the data in the table was deleted. Using the SENSOR database, first create the database and enable it for forward recovery:

```
DBM CREATE DATABASE SENSOR ON D
DBM UPDATE DATABASE CONFIGURATION FOR SENSOR
USING LOGRETAIN ON
```

The "log retain enable" (LOGRETAIN) parameter is set to ON to enable the database for roll-forward recovery. This command tells the database that all transactions should be recorded in log files.

The files are stored in the directory given in the "Path to log file" parameter of the database configuration. This directory can be changed by providing a new path to the "Changed path to log file" (NEWLOGPATH) parameter. Initially, it does not contain a value. After changing the LOGRETAIN parameter, the current configuration values of a database can be checked by issuing the following command:

```
DBM GET DATABASE CONFIGURATION FOR SENSOR
```

The report returned by this command for the SENSOR database is contained in Figure 5.

The NEWLOGPATH parameter does not have a value, and the default "Path to log file" parameter is set to D:\SQL00003\SQLOGDIR\. All log files will be stored in this directory. Additionally, the LOGRETAIN parameter is set to ON. The parameters in parentheses are the ones used in the UPDATE DATABASE CONFIGURATION command.

The default "Path to log file" directory will be kept. However, with the database's LOGRETAIN parameter enabled, the database must be backed up before it can be used, so the log files start from a stable backup source file. (Roll-forward recovery works by first having a restored database and applying the transaction logs to this base.) The database can be backed up to the A: drive with the command:

```
DBM BACKUP DATABASE SENSOR TO A
```

After following the prompts, the database will be backed up and ready for use. Next, the table used to store the sensor data must be created. This can be accomplished by first connecting to the database and then issuing the CREATE TABLE command:

```
DBM START USING DATABASE SENSOR
DBM CREATE TABLE READINGS ( VALUE SMALLINT,
VTIME TIME )
```

When the START USING DATABASE command is issued, REXX connects the current OS/2 session to the SENSOR database. This database connection is maintained until the user issues the STOP USING DATABASE command. Using the DBM Command Line Interface, an OS/2 session can only have one database connection at a time, much like a regular application. Of course, a user can open up multiple OS/2

```
      when keyword = 'ALL' then
      do
        say ' '
        say 'REPORT for:  LISTMY 'nameof  keyword  namein
        say ' '
        call dbm "select creator, name, type, remarks from sysibm.systables"
      end

      otherwise
      do
        say 'Unexpected keyword "'keyword'". ',
            'Expect keywords:  "NULL, ALL".'
        SQLCODE = '-1'
      end

    end /* select on keyword*/
  end  /* when table */

  when nameof ='INDEX' | nameof = 'INDEXES' | nameof = 'INDICIES' then
  do
    Select
      .
      .
      .
    end /* select on keyword */
  end  /* when index */

  when nameof ='COLUMN' | nameof = 'COLUMNS' then
  do
  Select
      .
      .
      .
  end  /* select when keyword  */
  end  /* when column */
  otherwise
  do
    say '                                                    '
    say '   LISTMY  -- ---------------------- ----          '
    say '             |                        |           '
    say '               TABLES _ ------ --------           '
    say '             |          '- ALL-'       |          '
    say '             |                        |          '
    say '               INDEXES  -------------             '
    say '             |            ALL --------  |         '
    say '             |         '- ON tblname -' |         '
    say '             |                        |          '
    say '             |_COLUMNS ON tblname ____|          '
    say '                                                  '
    say ' Retry using the correct syntax.                  '
    say '                                                  '
    SQLCODE = 100
  end

end /* Select on nameof */

rc = SQLCODE

return rc
```

*Figure 4: Source listing for LISTMY.CMD (continued from page 120)*

sessions and have separate connections to the same or different databases.

The CREATE TABLE command creates the READINGS table with two columns. The VALUE column stores the value of the reading, and the VTIME column stores the time the reading was taken.

To simulate data from the sensor, issue the following commands:

```
        Database Configuration for Database SENSOR

Max. no. of active applications   (MAXAPPLS) = 8
Max. DB files open per appl.      (MAXFILOP) = 20
Max. files open per appl.      (MAXTOTFILOP) = 255
Default application heap        (APPLHEAPSZ) = 3
Application agent heap           (AGENTHEAP) = 2

Max. storage for lock list        (LOCKLIST) = 25
Percent of lock list allowed      (MAXLOCKS) = 22
Deadlock check time interval     (DLCHKTIME) = 10000

Buffer pool size                  (BUFFPAGE) = 250
Database heap                       (DBHEAP) = 3
SQL statement heap                (STMTHEAP) = 64
Sort list heap                    (SORTHEAP) = 2

No. log records for checkpoint     (SOFTMAX) = 100
Log file size                     (LOGFILSIZ) = 200
No. of primary log files         (LOGPRIMARY) = 3
No. of secondary log files       (LOGSECOND) = 2
Path to log file                             = D:\SQL00003\SQLOGDIR\
Changed path to log file         (NEWLOGPATH) =
First active log file                        =
Next active log file                         =

Database attributes                  (DBATTR) = 11
  Copy protect enable           (COPYPROTECT) = ON
  Log retain enable               (LOGRETAIN) = ON
  User exit enable                 (USEREXIT) = OFF
  Auto restart enable           (AUTORESTART) = ON

Database State                               = 1
  Database is consistent                     = YES
  Backup pending                             = NO
  Rollforward pending                        = NO
  Log retain status                          = OFF
  User exit status                           = OFF

Database country code                        = 1
Database code page                           = 437
Database release level                       = 0x0300
```

*Figure 5: Database configuration for the SENSOR database on a database server*

```
DBM INSERT INTO READINGS (VALUE, VTIME) VALUES
(10, '12:10:30')
DBM INSERT INTO READINGS (VALUE, VTIME) VALUES
(11, '12:10:33')
DBM INSERT INTO READINGS (VALUE, VTIME) VALUES
(67, '12:10:38')
```

There can be more inserts; with a real-time sensor, there would be. The three inserts are illustrative for this example.

Next, check to insure that these data rows were stored by issuing a SELECT on the table:

```
DBM SELECT * FROM READINGS
```

A report will be returned, as in Figure 6. Before simulating a loss of the data, first record the current system date and time. This function can be performed by issuing the DATE and TIME commands from the OS/2 prompt and recording the values. This example assumes the date and time are 02-21-1992 and 14:24:30.00, respectively.

To simulate data loss, enter:

```
DBM DELETE FROM READINGS
DBM STOP USING DATABASE
```

The first command deletes all the data rows from table READINGS, and the second command disconnects the command line from the database. (A check to make sure the rows were deleted can be performed by issuing a select on the READINGS table before disconnecting from the database. A DROP TABLE could also have been used to erase the table and rows.) Whatever the method, the data is now lost. It could have been an intentional (yet later regrettable) or an accidental loss. The database is in a consistent state and can continue to be used, but from the user's perspective, it is incorrect since the needed data rows are no longer there.

All is not lost. Since the database was enabled for roll-forward recovery and all the transactions were logged, the data rows can be recovered by issuing the commands:

```
DBM RESTORE DATABASE SENSOR FROM A TO D
DBM ROLLFORWARD DATABASE SENSOR TO 1992-02-21-
14.24.30.00 AND STOP
```

The first command restores the SENSOR database from A: to D: (or whatever drive the database was created on) and places the database in a

"roll forward pending" state; the database cannot be used until it is rolled forward. The restored database replaces the existing one, and as such, it does not contain the READINGS table.

The second command performs the actual roll forward. The date and time were combined to form the timestamp in ISO format. This timestamp is required whether the database is rolled forward by the DBM Command Line Interface or by other interfaces. Thus, the logs are rolled forward to the date and time recorded earlier. The AND STOP parameter completes the roll forward, backs out any incomplete transactions, and enables the database for use. A report similar to Figure 7 will appear after the roll forward action completes.

According to the report, the database had only two archive log files. If additional log files were expected, the name of the next log file would appear after the "Next archive log file" parameter in the report. The DBM allows users to roll the logs forward in stages, so long as the STOP parameter is not specified. Thus the log files can be copied from a backup medium to the log file path before continuing with the recovery process. This process can be accomplished either manually or by using the userexit function.

Here comes the real test. To insure that the deleted data is back in the READINGS table, enter:

```
DBM START USING DATABASE SENSOR
DBM SELECT * FROM READINGS
```

If the roll-forward recovery was successful, the same SELECT report generated earlier should appear.

## OTHER DATABASE MANAGER INTERFACES

Extended Services comes with other user interfaces to the DBM. Query Manager has complemented the DBM since its initial release, and it continues to work adeptly with databases as a query and application development interface. Three additional interfaces make their debut in this release. Under the DBA Tools banner, they are the Directory Tool, Configuration Tool, and Recovery Tool. Almost every action a user can perform with these tools can also be done with the DBM Command Line Interface, yet these distinctive user interfaces provide additional information to database administrators and application developers.

```
VALUE  VTIME
------ --------
    10  12:10:30
    11  12:10:33
    67  12:10:38

 3 record(s) selected.
```

*Figure 6:* SELECT * FROM READINGS

```
Roll Forward Status for database: SENSOR

Next archive log file          =
First archive log file         = S0000001.LOG
Last archive log file          = S0000002.LOG
Last committed transaction     = 1992-02-21-14.23.49.000000
```

*Figure 7: Roll forward status for the SENSOR database*

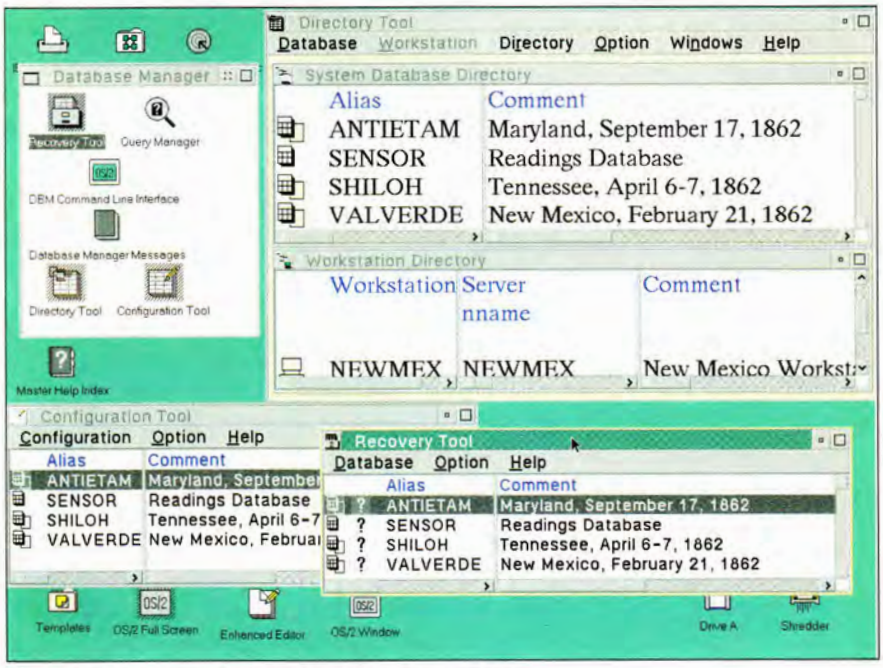



*Figure 8: The Directory, Configuration, and Recovery Tools*

For example, an application developer who needs to configure a workstation environment with clients and servers for testing purposes can use the Directory Tool to catalogue

workstations and remote databases. Lists of the workstations and databases can be shown in the same window along with their attributes. The Configuration Tool presents a list of databases and allows users to change resource parameters on the selected database or for the DBM itself. The Configuration Tool is the only interface available to configure a remote database. The Recovery Tool provides prompts to back up, restore, restart, and roll forward the databases a user selects. Figure 8 shows each of the tools. The earlier forward-recovery example could be accomplished by using Query Manager and the Recovery Tool.

```
@echo off
REM  Catalog the Department Server and Database
call DBM catalog netbios node antietam remote maryland adapter 0
  if not errorlevel  0  goto quit
call DBM catalog database burnside at node antietam with "Bridge db"
  if not errorlevel  0  goto quit
call DBM update database manager configuration using comheapsz 20
exit
REM  Error branch
:quit
echo An error has occurred.
exit
```

*Figure 9: Source listing for* `MYSETUP.CMD`

Alternatively, an easy way to configure a number of workstations would be to write a CMD file with the appropriate commands and execute them on each workstation. For example, the OS/2 command file MYSETUP.CMD in Figure 9 catalogues a workstation and database and increases the "Communication Heap Size" parameter for the DBM at the local workstation.

## CONCLUSION

The DBM Command Line Interface provides an easy and flexible way of accessing Database Manager functions. Commands can be executed individually, or they can be combined to form more powerful command files. Developers and administrators can use command files, in conjunction with other programs, to supplement their applications. For example, an application that executes a program at a specified time can be used to run a command file that backs up a database to a tape drive at night when everyone's away. A complex program that calls the DBM backup API is not needed. The DBM Command Line Interface can be used instead.

## ACKNOWLEDGMENTS

The authors would like to thank Lee Atkinson, Garland Bullock, Lavena Chan, and Steve Curry for their time and assistance.

**Theodore Shrader**, *IBM Personal Systems Line of Business, 11400 Burnet Rd., Austin, Texas 78758. Shrader is a senior associate programmer who has worked on database tools for the DBM product. He joined IBM and the database group in 1989 and has worked in graphical application development since then. He has a B.S. in computer science from the University of Texas at El Paso. (Go Miners!)*

**Joel Dunn**, *IBM Personal Systems Line of Business, 11400 Burnet Rd., Austin, Texas 78758. Dunn is a staff programmer for IBM Austin. He joined IBM in May 1982 and has worked on user-interface tools for the OS/2 DBM project since 1987. He received a B.S. in computing science from Texas A&M University. (Gig 'em, Aggies!)*

## REFERENCES

*OS/2 REXX Commands Reference* (IBM Doc. S01F-0271; 01F0271)

*Procedures Language 2/REXX Reference* (IBM Doc. S01G-6268-00; P10G6268)

*Procedures Language 2/REXX User's Guide* (IBM Doc. S10G-6269-00; P10G6269)

The following are part of the Extended Services documentation:

*Guide to Database Manager* (IBM Doc. S04G-1013; 04G1013)

*Structured Query Language* (SQL) Reference (IBM Doc. S04G-1012; 04G1012)

On-line reference files ESCMDREF.INF and DBMSG.INF

Hardware

# Making OS/2 Run Across Industry PCs

*by James Gillig, Craig Bender, and Chris von Koschembahr*

*IBM is working to ensure that OS/2 2.0 runs successfully on both IBM and non-IBM PCs. In many cases, this involves working directly with other PC manufacturers. The rapid growth of the personal computer industry in its first 10 years has resulted in the worldwide formation of hundreds of PC manufacturing companies, also known as original equipment manufacturers or OEMs. This growth has also resulted in continually improving PC price and performance, a greater variety of PC brands and models, and local area networks (LANs) that can connect a heterogeneous group of computers from multiple manufacturers. To serve the growing PC industry and meet customer needs, OS/2 is intended to run on all Intel-compatible 386SX and higher-level systems throughout the industry. This article highlights some of the ways in which OS/2 and its communications, LANs, and database extensions support the PC manufacturing industry and its customers.*

## PC MANUFACTURERS

Providing and supporting a single IBM version of OS/2 that can run on multiple hardware platforms reduces the need for separately adapted versions of OS/2 for different PCs. A single version also offers consistency for users moving between workstations. In OS/2 versions prior to 2.0, IBM tested, marketed, and supported OS/2 only for IBM hardware. Before the release of version 2.0, other PC manufacturers received OS/2 adaptation kits through Microsoft Corp. Because of this arrangement, PC manufacturers were dependent on Microsoft to provide the kits, and updated versions of OS/2 did not become available to all users at the same time.

Most businesses use hardware from several PC manufacturers. Businesses desire a common look and feel for all workstations, as well as ease of use and maintenance. These goals are best met by a single operating system that can be supported across a variety of hardware. Businesses have become populated with PCs from various manufacturers for a number of reasons. A company or government agency may have a low-cost bid policy, or it may have specific technical requirements that narrow the selection to certain computers or include a package of hardware, software, and support services. A company may also have a dual sourcing policy, requiring that purchases be made from more than one manufacturer. This ensures competition between manufacturers who are vying for the customer's business, reduces dependency on a single supplier, and provides access to a larger base of technology. One company may also acquire or merge with another and, as a result, inherit different computer hardware.

IBM has taken a number of steps to better support OS/2 on different manufacturers' hardware. A 10,000-square-foot laboratory has been created at its Boca Raton, Fla. development center to test OS/2 2.0 and its extensions on non-IBM hardware; hundreds of different PCs can be tested in this laboratory at one time. IBM representatives have also spoken with hardware manufacturers about supporting OS/2 software on their hardware, programming native OS/2 code to operate on non-IBM hardware platforms, and acquiring device drivers from SCSI and Super VGA vendors. IBM is also exchanging hardware and software with other manufacturers and performing tests to ensure compatibility between systems and software.

*Jim Gillig*

*Craig Bender*

*Chris von Koschembahr*

## USER SERVICE AND SUPPORT

A user can set up numerous combinations of computer hardware, input and output devices, and software. While IBM and other manufacturers test many computer brands, models, and hardware configurations running under OS/2, it is impossible to cover the seemingly limitless combination of Intel processors, processor speeds, memory and DASD storage configurations, disks and their controllers, video adapters and displays, attached devices, I/O buses, BIOS versions, and software configurations. To help users, an OS/2 README file, telephone support line, and OS/2 bulletin boards are available.

*OS/2 OEM test lab in Boca Raton, Fla.*

The OS/2 README file, installed as part of OS/2, includes service and support information, OS/2 installation considerations, Workplace Shell user tips, specific application considerations with OS/2, performance considerations, Windows 3.0 considerations when running WIN-OS/2 sessions, video and graphics support considerations, and hardware considerations for OEMs and independent hardware vendors.

Users who encounter problems with OS/2 2.0 on their system configuration can call the toll-free IBM OS/2 support line for limited technical help or optional extended support. Other sources of help for OS/2 include bulletin board systems (BBSs) such as the OS/2 BBS (call 1-800-547-1283 to register), an IBMOS2 forum on CompuServe (call 1-800-848-8199 and ask for representative 239), the IBM National Support Center BBS (call 1-404-835-6600), Prodigy™ (call 1-800-PRODIGY), and other regional OS/2 BBSs (call 1-609-596-1267 for locations).

## DESIGNING OS/2 FOR PC MANUFACTURERS

The design of OS/2 2.0 supports the broader industry of PC manufacturers in several ways. OS/2 on-line presentation information, help information, system messages, and visible text have been made more generic; because text no longer contains specific references to IBM or OEM hardware, system information will not confuse users with inaccurate references.

OS/2's executable code has been made more generic. It is no longer hard coded to recognize and handle only IBM PS/1™- or PS/2-specific systems or conditions. The new code should execute on any system compatible with Intel 386SX or higher platforms. In addition, OS/2 fully supports bus architectures for EISA, ISA, and MCA systems.

*"...The line of communication between the Compaq and IBM development and test groups allowed Compaq's customer requirements to be smoothly incorporated into the standard IBM product and thoroughly tested across the COMPAQ product line."*

*—Mike Clark, vice president for systems marketing, Compaq Computer Corp.*

The architecture of OS/2's DASD device driver has been improved to simplify the replacement and addition of device support through a layered adapter device driver (ADD). In OS/2 1.3, a base device driver was a large and single module such as a disk or SCSI driver; in OS/2 2.0, the DASD device driver has been structured as smaller replaceable layers.

## DEVICE DRIVER ADD

OS/2 2.0 has beed improved so a DASD device driver can now be written in two layers. The higher level, the DASD device manager, communicates with the kernel, while at the lower level the adapter device driver (ADD) communicates with the adapter device. Adapter device support for non-IBM DASD and SCSI equipment can be provided through an ADD. In addition, an optional filter ADD, which can compress, encrypt, or otherwise process I/O data before communication between a device and requesting program begins, can be used between a device manager and its ADD.

OS/2 provides two device managers, OS2DASD.SYS and OS2SCSI.SYS, for DASD and SCSI devices. The interface for an ADD module includes direct call commands issued by a device manager or filter ADD through the ADD's registered entry point as well as a global pointer to the ADD's I/O request block. Direct call commands that can be made to an ADD include:

- `CONFIGURATION`
- `GEOMETRY`
- `FORMAT`
- `DEVICE_CONTROL`
- `UNIT_CONTROL`
- `EXECUTE_IO`
- `UNIT_STATUS`
- `ADAPTER_PASSTHRU`

In addition, `DevHlp` services are provided. A user can install and load an ADD through the new CONFIG.SYS statement `BASEDEV=name of ADD file`, where the named file is known as an adapter driver. Device drivers loaded with the old `DEVICE=name of driver file` statement are known as installable drivers.

## OS/2 COMPATIBILITY TESTING

The compatibility test process is an important part of making OS/2 2.0 and its extensions available on all PC manufacturers' hardware platforms. In the IBM compatibility testing lab, numerous manufacturers' hardware is tested for compatibility with OS/2 products. As the test process evolved, it became evident that a mechanism must be established to ensure accurate testing. The original Test Kit consisted of the object code for OS/2 2.0 and its extension products, including Extended Services (ES) 1.0, LAN Services (LS) 2.0, and their associated publications, problem determination aids, and technical support. The Test Kit was periodically updated as stable intermediate code drivers become available for testing. Each Test Kit included a newsletter describing new functions and inclusions in the Test Kit package.

*"...The IBM/AST test and support agreements gave our customers the confidence that calling AST or calling IBM would produce the same complete answers to their questions. It's exactly what AST's customers wanted."*

*—Michele Smith, senior marketing manager for software products, AST Research Inc.*

*Testing OS/2 on OEM notebook PCs*

The primary advantage of the Test Kit was that it delivered early code releases of OS/2 and its extensions to OEMs. While manufacturers were invited to loan IBM new and announced models of their products, due to the expense involved some companies were only able to send a few models or none at all. The Test Kit allowed these manufacturers to test their entire product line in-house for OS/2 compatibility. It also allowed PC manufacturers to test unannounced models directly and to claim OS/2 compatibility as soon as their machines become publicly available. OEMs could also determine the compatibility of their or others' adapter cards. The Test Kit's early drafts of OS/2 publications and problem determination aids also benefited other manufacturers, helping them to develop strategies for determining the compatibility of their hardware with OS/2 products. These drafts also improved manufacturers' familiarity with OS/2 products and their features and functions. Combined with the problem determination aids in the Test Kit, they enabled manufacturers to improve customer service and support.

*"We're committed to providing an OS/2 solution; an increasing number of our large accounts are telling us this is a requirement."*

*—Judith Bitterli, director of sales, CompuAdd Computer Corp.*

A key to the success of the Test Kit was the technical support provided to all Test Kit participants, who could call IBM's technical development team for product usage help and problem determination. To tackle potential problems, the test environment was recreated at IBM on

## OS/2 ES/LS TEST KIT PARTICIPANTS

Acer America™

Advanced Logic Research™

Apricot Computers™

AST Research™

AT&T Data Systems Group™

Club America Tech.™

Commodore International

Compaq Computer™

CompuAdd™

CSS Laboratories™

Dell Computer™

Digital Equipment Corp.™

Dtk Company™

Epson Portland™

Everex System™

Grid Systems™

Hewlett Packard™

Intel™

Leading Edge Productions™

Memorex Telex™

Mitac International™

Modern Computer™

NCR™

NEC Technologies™

Netframe Systems™

Northgate Computer Systems™

Olivetti Advanced Tech.™

Parallan Computer™

Reply™

Siemens Nixdorf International™

Tandon™

Tandy Electronics™

Tatung Co. Of America™

Toshiba America™

Tricord Systems™

Wyse Technology™

*Table 1: OS/2 ES/LS Test Kit participants*

identical or technically similar models. The test procedure has resulted in changes to both OS/2 code and manufacturers' hardware.

Regular conference calls were established with many Test Kit participants. The calls typically consisted of discussion of test progress, test issues, and outstanding problem status. The calls also provided an opportunity to exchange news of upcoming products or marketing events.

Many PC manufacturers, including those in Table 1, joined the Test Kit program.

IBM makes specific claims of OS/2 compatibility only for those models that have been successfully tested at its test lab, or by OEMs using the Test Kit. These claims are made public on bulletin boards, including Prodigy and CompuServ. IBM is also publishing the names of participants in the Test Kit program and suggests that customers contact those manufacurers directly concerning their compatibility with OS/2.

The Test Kit has evolved to support OS/2 2.0 compatibility testing by other PC manufacturers. Available under license from IBM, the OS/2 2.0 version 1.0 Test Kit includes program tests such as acceptance level and stress tests for OS/2 2.0 on OEM hardware. Acceptance level tests, which check major operating system components, begin with an OS/2 installation test. Once the system has been properly installed and tested, the rest of the tests run independently in sequence. All acceptance level tests can be run in less than a day. Stress tests, which are run only after all acceptance level tests are successful, are more demanding, testing major functions concurrently over six to eight hours.

An OEM that has completed OS/2 testing can send results to IBM for compatibility evaluation. Passing systems are added to IBM's list of OS/2 2.0-compatible systems, which is posted on bulletin boards including Prodigy and Compuserv. IBM suggests that customers also contact PC manufacturers about OS/2 compatibility.

*Testing OS/2 on OEM workstations*

## PREINSTALLATION

OS/2 2.0 is now preinstalled and shipped ready to run on certain IBM PS/2 models. In addition, IBM is working with OEMs who want to make preinstalled OS/2 available on their manufacturered PC systems.

> *"...Testing by Reply and IBM ensures that Reply clients are getting systems that are compatible and that work in many environments and applications."*
>
> *—Steve Petracca, president amd CEO, Reply Corp.*

The preinstallation system has all functions and code, including, device drivers, printer drivers, and utilities, from the OS/2 installation disks already installed on the machine. The elimination of installation disks saves money for both IBM and the customer; if a customer wishes to purchase installation disks, they are available separately.

Three preinstallation utilities, also available to OEMs, help the user take advantage of system versatility:

- The "Configure" utility allows the user to modify the display, mouse, keyboard, and printers to fit individual needs.
- The "Create Utility Diskettes" utility helps the user create a set of OS/2 bootable utility diskettes.
- The "Selective Uninstall" utility helps the user delete unneeded code from the hardfile, creating more available storage. Users can delete unused printer drivers, unneeded system drivers, or any other optional operating system parts.

*"IBM and ALR have entered into an agreement to compatibility test and support OS/2 2.0 on ALR PC systems...we will continue to ensure compatibility of OS/2 and ALR systems."*

*—Vic Sangveraphunsiri, vice president of systems engineering, Advanced Logic Research*

## SUMMARY

Businesses and individual users have a wide range of PCs from different manufacturers. Some may be configured as stand-alone desktop or portable systems and others as network clients or servers. To satisfy the need for a single, powerful operating system, OS/2 2.0 is designed to run on all PCs based on Intel 386SX and higher platforms. This makes the most recent OS/2 version universally available and provides a consistent and compatible multitasking operating system environment for OS/2, DOS, and Windows applications. IBM, OEMs, and independent hardware and software vendors are working together to make OS/2 available to all of the PC industry and its customers.

## ACKNOWLEDGMENTS

We would like to thank Cynthia Erdman of IBM for her early leadership in working with OEMs.

**Craig Bender** *is a senior programming consultant on the Corporate Technical Strategy Development Staff at IBM in Armonk, N.Y. Bender joined IBM in 1982 as a database administrator at Federal Systems Division, Owego, N.Y. In 1985, he transferred to Austin, Texas, where he designed and managed OS/2 database and communications subsystems. He also developed and implemented IBM strategy for supporting IBM OS/2 products on non-IBM PCs. He received a B.S. in computer and information science from the University of Delaware.*

**James Gillig,** *IBM Personal Systems Programming Center, 1000 N.W. 51st St., Boca Raton, Fla. 33429. Gillig is a member of the OS/2 architecture department and is currently chair of the OS/2 Architecture Board. He presented IBM OS/2 strategy to different PC manufacturers and worked with the IBM OS/2 Development Team to make OS/2 run on PCs throughout the PC industry. He holds an M.S. in computer science from the University of Missouri and a B.S. in mathematics from Western Illinois University.*

**Christopher von Koschembahr,** *IBM Personal Systems Programming Center, 11400 Burnet Rd., Austin, Texas, 78758. Von Koschembahr is a development programming manager who joined IBM in 1983, developing test-design automation software. He has led software tests in VM/SP Shared File System and OS/2 Extensions. He is now manager of the PC Manufacturers Management and Test department. Von Koschembahr holds a B.S. in electrical engineering and computer science from Duke University.*

# Dramatically Reduce Your Development Time

# The Natural Migration

IBM is offering 5 day, intensive, "hands-on" workshops that facilitate the task of migrating your application to the OS/2® operating system. The workshops combine classroom lectures with extensive lab sessions. Attendees are assigned to workstations and provided with the hardware and the technical assistance needed to port their application. A significant portion of the migration should be completed during the week and attendees will leave with the knowledge necessary to complete the job.

The following is a list of OS/2 2.0 workshops for 1992 now being offered in West Palm Beach, Florida

**DOS to OS/2 using VIO**

**OS/2 (16-Bit) PM® to OS/2 (32-Bit) PM**

**Windows 3.0™ to OS/2 (32-Bit) PM**

**OS/2 using VIO to OS/2 (32-Bit) PM**

For Schedule, Registration and Fee Information Call:

1-407-982-6408

Reserve your place in the OS/2 2.0 Workshop now being offered in Austin, Texas

**OS/2 Database Manager Performance Workshop**

For Schedule, Registration and Fee Information Call:

1-512-823-2359

IBM

® IBM, OS/2 and Presentation Manager are registered trademarks of International Business Machines Corporation

© IBM Corp. 1992

International

# EMEA Developer Assistance Program Update

*Vicky Gallop*

*by Vicky Gallop*

Now out of its pilot phase, the Europe/Middle East/Africa IBM Developer Assistance Program (EMEA DAP) is growing quickly; approximately 50 companies join each week. Many countries have nearly reached their target number of developer signups and are eager for more to join.

All 10 lines of the 24-hour on-line network are frequently in use as DAP members, or DAPpers, download files, exchange e-mail, and answer OS/2 programming queries.

The first non-U.K. European DAP workshop was held the first week of June in Paris, France. Workshop participants voted it a great success, noting that it saved development time by speeding up migration of applications to OS/2. Additional 1992 workshops are planned for Israel, Italy, Scandanavia, and South Africa.

**EMEA DAP MEMBER COUNTRIES**

| | | |
|---|---|---|
| Austria | Ireland | Spain |
| Belgium | Israel | Sweden |
| Denmark | Italy | Switzerland |
| Finland | Netherlands | Turkey |
| France | Norway | U.K. |
| Germany | Portugal | |
| Iceland | South Africa | |

*Table 1: EMEA DAP member countries*

## EMEA DAP MID-YEAR TOTAL MEMBERSHIP

The countries shown in Table 1 have a total mid-year membership of 750.

## EMEA DAP SERVICES

The EMEA DAP provides a variety of services to its members, including:

- Answers to technical questions on all aspects of writing OS/2 applications provided by a team of skilled OS/2 programmers, available 24 hours a day every day
- In-depth technical training at monthly workshops held throughout Europe
- Savings on hardware and software through the registered ISV program
- Free advertising for developers' software in the *OS/2 Applications Solutions Directory*
- Copies of OS/2 CSD code
- Technical books and other developer aids.

## AUSTRALIA/NEW ZEALAND DAP

The OS/2 Developer's Seminar, scheduled for August in Australia and Signapore, brings the highly successful OS/2 Technical Conference to Australia and Singapore. A representative from Pryda of Australia will speak. Pryda Computa-Roof™ v.3 is a unique success story. With a two million dollar investment on fewer than 150 licenses, Pryda succeeded and profited, winning an Australian Software

Design Award for its product, which is used to produce cost estimates and production drawings for nailplated, timber-trussed roofs. Pryda's marketing approach is simple: identify the need, understand the problem, and develop the best solution. In its 15 years, Computa-Roof has evolved from using programmable calculators to utilizing the full power of OS/2 2.0.

***OS/2 Australia/New Zealand Bulletin Board: Free Access***

Phone: 61-2-241-2466
300-9600 bps V.32 mnp5
Multiple lines, 8 data bits, no parity, 1 stop bit

***For DAP Registration Contact:***

IBM Australia/New Zealand DAP
Rohaini Cain
Phone: 61-2-354-7684
FAX: 61-2-354-7766

## INTERNATIONAL SUBSCRIPTIONS

International subscriptions are U.S. $55.95 for Canada and Mexico; others are U.S. $55.95 surface, $69.95 airmail. Subscribers outside of the U.S. and Canada may call (708) 647-5960 or fax (708) 647-0537.

International bulk subscriptions are also available. Price quotes can be obtained from the publisher. Call Kathy Henry at Miller Freeman Inc. at (415) 905-2260 or fax (415) 905-2233, Pacific Standard Time.

IBM employees may subscribe through SLSS; additional copies of the current issue and some recent back issues may be ordered through PUBORDER. The IBM order number is G362-0001.

## ACKNOWLEDGMENTS

Information on Australia/New Zealand DAP was submitted by Clive Astle.

**Vicky Gallop**, *IBM U.K. Ltd., Mountbatten House, Basing View, Basingstoke, Hants., RG21 1EJ, U.K. Gallop is a marketing specialist in the OS/2 Group of the EMEA Personal Systems Centre. Before working at IBM, Gallop worked as a statistician. She then worked for several computer consulting companies before joining IBM. Gallop holds a B.Soc.Sc degree in economics from the University of Birmingham.*

Expanding its support for OS/2 software developers, IBM has opened an on-line CompuServe Developer Assistance Program for system integrators, MIS professionals, consultants, educators and students, industry analysts, and programmers worldwide. To enroll, sign on to CompuServe and enter GO OS2DAP.

# Interview: Bob Orfali and Dan Harkey

*Bob Orfali and Dan Harkey, authors of* Client/Server Programming with OS/2 2.0, *recently spoke with* OS/2 Developer *editor Dick Conklin. Their 1,100-page book, now in its second edition, is often considered the bible of client/server programming on the OS/2 platform.*

*"An 1,100-page book is hard to digest without a bit of humor."*

*Developer:* What about your book's new look—who are the cartoon characters on the cover?

Bob: We call them the "Mon people." They show up from time to time in the book to make wisecracks, or sometimes just wise points. They help us tell the OS/2 client/server story. An 1,100-page book is hard to digest without a bit of humor.

*Dan Harkey and Bob Orfali*

*Developer:* I noticed that your book got fatter. What's new in the second edition?

Bob: Six hundred new pages, actually. The book was a major overhaul of the first edition that took us over 10 months to complete. It was a big effort to keep up with the advances in OS/2, but we're proud of the results.

Dan: There were changes everywhere. C Set/2™ helped a lot with the 32-bit OS/2 conversion. The new OS/2 2.0 semaphores and flat memory model are nice to work with. We added TCP/IP™, the NetWare Requester™, and CPI-C/APPN™ to the communications section —now you can see how they compare with NetBIOS™, Named Pipes™, and LAN Server. In the Database section we added DDCS/2™ and the new roll-forward feature. But the most exciting new addition is the section on OOUI clients.

*Developer:* OOUI?

Dan: OOUI stands for object-oriented user interface. The Workplace Shell and Macintosh are OOUIs; Windows, Motif™, and OS/2 1.X are GUIs. It turns out that OOUI is what makes OS/2 2.0 a great client platform but it introduces a whole new way of programming.

*Developer:* What kind of new programming?

Dan: We used SOM and the Workplace Shell Class libraries to create our OOUI front ends. The last 200 pages of the book are a crash course in object-oriented programming using SOM/WPS classes. It took us a long time to figure out how to make it all work in a client/server environment.

Bob: But it was well worth it. We gave our Club Med example an OOUI facelift. You can see the dramatic improvement over the first edition's Club Med, which was a Windows-vintage GUI. Club Med is now truly integrated into the Workplace Shell. It graphically

demonstrates how the Workplace Shell can be used to create a new generation of super-client software. Our Club Med server provides a persistent object store for Workplace Shell folder objects using a new `doRPC` SOM/WPS class we created.

*Developer:* Is SOM/WPS a viable programming platform?

Dan: Yes, once you get the hang of it. As with any platform, it's important to find some working code you can build on. We had problems because we had to build our environment from scratch. Nobody had code that tied the SOM/WPS to a client/server environment. The five new SOM/WPS classes we created in the book will give our readers a head start.

Bob: I think OOUIs will provide the killer apps that differentiate OS/2. Once you start using one, there's no going back to GUIs. OOUIs are really fun—the problem is that very few developers know how to create them on OS/2. We hope our book will encourage people to write new OOUI applications. There's a steep learning curve, but it's worth it. It was the same with the Macintosh™ a few years ago.

*Developer:* Who was the winner of this edition's BLOB Olympics?

Bob: Believe it or not NetBIOS...somebody fed it lots of vitamins this time around. CPI-C, TCP/IP, and Named Pipes over LAN Server ran a close race for second position. Named Pipes over the Novell Requester for OS/2 was the worst performer. Of course, you can probably do better with NetWare's native IPX/SPX™.

Dan: I believe NetBIOS was totally rewritten to run at Ring 0 and interface directly with NDIS. It's a screamer.

*Developer:* Are transaction servers (fat servers) still faster than database servers (fat clients) in your TP1 races?

Dan: Of course, some things never change. Interestingly, dynamic SQL improved a bit. It's still a distant third—but at least it shows up on the charts.

*Developer:* Where do you see OS/2's role in client/server computing?

Bob: We think OS/2 provides the ideal platform for both the client and server in the world of PCs, DOS, Windows, and OS/2. PC LANs constitute the largest segment of the client/server market. By the end of this year, over 50% of business PCs will be on LANs. That's a huge market. OS/2 provides a solid client platform and integrates the visual information on the desktop-using OOUIs. OS/2 is a great server platform for PCs, but it will encounter some stiff competition.

*Developer:* Where's the competition coming from?

Bob: NetWare™ 3.X and UNIX today, and Windows-NT™ next year. But you'll have to read the first 200 pages of our book to get the full story. We wrote that part of the book for a general audience—architects, system analysts, managers, and other decision makers. In those 200 pages, we explain in detail what OS/2 has to offer as both a client and a server platform. It's a story that needs to be told.

*Developer:* Why did you use OS/2 for your client/server book?

*"We hope our book will encourage people to write new OOUI applications. There's a steep learning curve, but it's worth it."*

Dan: It was important to have a book that went past the armchair discussion of client/server and supported the architectural explanations with working code on which we could run experiments like fat clients vs. fat servers, GUIs vs. OOUIs, and the BLOB Olympics.

Bob: And OS/2 gave us a very advanced platform on which to develop that code. The best client/server software requires threads, multitasking, memory protection, support for huge objects (BLOBs), multimedia, database, RPCs, and so on. OS/2 has it all now. It's a perfect platform from a client/server developer's viewpoint.

*Developer:* Will there be a third edition of the book?

Dan: Of course; there will always be a need for a third edition. OS/2 is not going to sit still. The question is, will Bob and I have the energy to create another one of these monster books? You know, it's a huge undertaking.

Bob: Well, I already see the need to cover C++, NetWare 4.0, digital video multimedia, DCE, symmetric multiprocessing servers, distributed objects... OS/2 and client/server technology are both moving at very rapid pace. Client/server brings everything together —multitasking, objects, OOUIs, and RPCs— and makes for very fat books. We'll probably write a third edition if this one is successful.

*Developer:* Is there a demand for OS/2 books?

Bob: Definitely. I've got to admit that it was very hard to get an editor interested in publishing our first edition, and if it wasn't for your help it wouldn't have been published. But as you know, it turned out to be a hit and was on the computer best-seller list. We expect this edition to do even better. But we still need to get the word out that the book exists and is sold in major bookstores.

Orfali, Robert, and Dan Harkey, *Client/Server Programming with OS/2 2.0*, 2d ed. New York: Van Nostrand Reinhold, 1992. $39.95 (IBM Doc. G325-0650)

The book is available at bookstores or call (800) 842-3636 to order.


# GREEN-KEEPER

**(noun) 1. One who keeps the environment green. 2. A series of clip-and-save tips for keeping your business and home environments green.**

**#1**

## How to help your company go green.

From the boardroom to the mailroom to the lunchroom to the washroom, there may be more ways than you think to help your company become environmentally responsible:

- Recycled office paper
- In-house recycling programs
- Reduced or recycled product packaging
- Glassine or open window envelopes
- Compact fluorescent light bulbs
- Company coffee mugs
- Weekly departmental newsletters instead of daily memos

But how can you help your company make these changes? Where can you find energy-saving, recyclable, and recycled materials? How can your company recycle its own waste? And how can you do it at little or no expense—or even *save* money?

Try some of the suggestions in *Keeping Your Company Green*, written by Stefan Bechtel and the editors of Rodale Press. This how-to guidebook is packed with "simple, doable steps your company can take to help keep itself—and the world—green." The book also includes a Directory of Green Products and Services. For more information, contact Rodale Press at 1-800-441-7761.

*Permission granted by Rodale Press, Inc.*

*The Green-Keeper series is sponsored by the MFI Green Project of Miller Freeman Inc.*

*A SHIELD Series*
*Development Tool*

# Presenting the Ultimate Resource Editors...

# RESOURCE SHIELD

The Tool for Visually Creating, Editing, Prototyping and Translating all OS/2, Windows and Windows NT Resources.

The fast way to design the look and feel of your programs

Available for Windows NT soon

*Browse all your resources quickly using the visual resource browsers. Hundreds of resource templates get you started quickly. Resource translators allow you to translate resources between all supported platforms. Organize and manage your resources using powerful project views.*

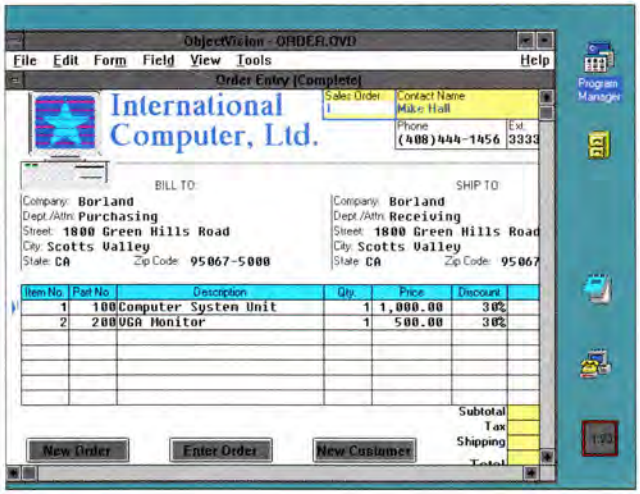
MICROSOFT® WINDOWS™ COMPATIBLE

**The Stirling Group**
Making it easy to make™
172 OLD MILL DRIVE • SCHAUMBURG, IL 60193 • USA • CALL 708-307-9197 • FAX 708-307-9340

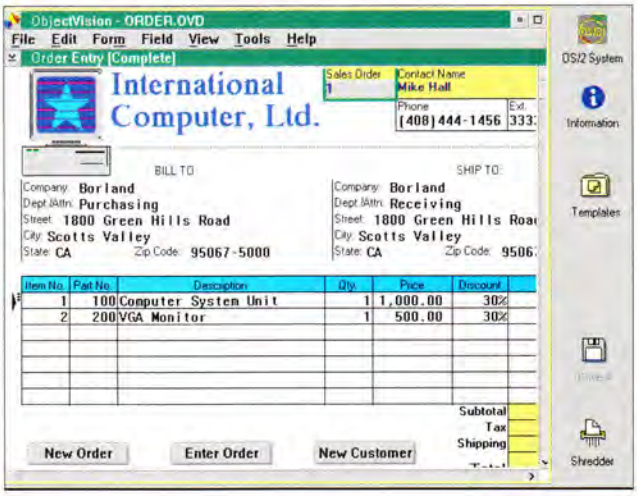
**1-800-3 SHIELD**
1-800-374-4353

New Version

# Q: What do Windows and OS/2 have in common?




ObjectVision for Windows




ObjectVision for OS/2

# A: Your ObjectVision applications.

Borland's ObjectVision® brings Microsoft® Windows and OS/2® users together. Only ObjectVision allows you to write an application *once* for use under both Windows and OS/2. And seamlessly access the same data, regardless of platform. That's because ObjectVision, the world's easiest tool for creating Windows applications, is now also available for OS/2.

## Visual programming makes it easy

Tied: #1 Application Software℠ Publisher in Customer Satisfaction*

Using revolutionary visual programming techniques, ObjectVision makes it easy for the people who know the business issues to develop and maintain their own graphical software applications without programming.

Creating Windows or OS/2 applications is as easy as A-B-C. Simply:

**A.** Draw your application interface using the convenient form tool.
**B.** Apply your business rules visually.
**C.** Connect your application to your existing database. Or create your own database.

In Windows and OS/2, ObjectVision makes it easy to create, modify and run your own interactive applications. And the OS/2 version gives you easy access to REXX as well as DLLs.

## Easy access to data

ObjectVision is the ideal front end to your data because it integrates information from different databases under one familiar, graphical user interface. Your ObjectVision applications can easily connect to information stored in Paradox,® dBASE,® Btrieve, Database Manager and ASCII formats.

## FREE runtime version

Because a runtime version is included *free*, it's easy to distribute your customized applications throughout your company. With ObjectVision, fully interoperable Windows and OS/2 applications are here today!

See your dealer today to get your own copy of ObjectVision for Windows or OS/2, OR call **1-800-331-0877, ext. 6620** to request your ObjectVision for Windows

## FREE TRIAL DISK

In Canada, call **1-800-461-3327**

**BORLAND**

*The Leader in Object-Oriented Programming*

G362-0001-15

Copyright © 1992 Borland International, Inc. All rights reserved. ObjectVision, Paradox and dBASE are registered trademarks of Borland International, Inc. OS/2 is a registered trademark of IBM. Microsoft is a registered trademark of Microsoft Corporation.
*1991 J.D. Power and Associates 1991 Computer End User Satisfaction Study for Application Software Publisher. Phase IV: Business End User Summary. Responses from Business End Users at 4,396 business sites. BI 1552